Adversarial and Stochastic Search for Mobile Targets in Complex Environments

A THESIS
SUBMITTED TO THE FACULTY OF THE GRADUATE SCHOOL
OF THE UNIVERSITY OF MINNESOTA
BY

Narges Noori

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
Doctor of Philosophy

Prof. Volkan Isler

February, 2016

Acknowledgements

I would like to first thank my advisor, Professor Volkan Isler for all his support and patience throughout my five years of studies. Thank you for trusting in me. I am grateful for the fun and the joy of working with you on such cool problems. I think the biggest chance of my academic life was ending up in Robotic Sensor Network lab and working on pursuit-evasion games. I could never guess that my PhD subject will be my teenage year's passion: Geometry.

I am thankful to my committee members Professor Ravi Janardan, Professor Stephen Guy and Professor Andrew Beveridge for invaluable feedback on my dissertation, and the endless support they have provided me in these years. Specially Professor Beveridge for his patience throughout these years especially in writing up one of our collaborative papers, "A pursuit-evasion toolkit" which is partially appearing in Chapter 3.

I cannot express my gratitude for my family and friends who their constant support made this journey possible and feasible for me. Two people are most influential in my successes: My mother, Fateme Khorsandi, who believed in me since I was a kid, and who supported me in every step I took. I couldn't ask for a better gift than you. Thanks for being the best. Then I'd like to thank Dr. Ehsan Noohi Bezanjani. Without you standing by me in every moment, I couldn't start and then finish my PhD. Thanks for being the best I could ask for.

I would like to express my deep thanks to my Dad, Mohsen Noori, for him trusting in me deep in his heart. Thanks for believing in me. Without you I couldn't stand in the place that I am today.

Thanks to my friends in Minnesota: You are my most precious achievements of these years. I leave part of heart behind me in Minnesota in my sweet memories. Special thanks to Zahra Sohrabpour and Zeinab Takbiri, Thanks for being you, the way you are, for all the runs we did together; and also to teach me how to drive, no one trusted in me except you. Thanks to Pantea Moghimi, Roshanak Navvab, Fateme Serajzade, Sanaz Khosravani and Dr. Maral Mousavi.

Thanks to my lab-mates, I couldn't ask for better colleagues than you. Professor Pratap Tokekar, Dr. Joshua Vander Hook, and Patrick Plonski. I experienced a new type of fun in my life: dinners after long experiment days. Thank you for being there, for your support and patience.

Dedication

To mom for her support and never-ending love, for teaching me to be strong and independent...

To my love Ehsan: I took your hands and dived in; It was all amazing as you promised...

Abstract

A new era of robotics has begun. In this era, robots are coming out of simple, structured environments (such as factory floors) into the real world. They are no longer performing simple, repetitive tasks. Instead, they will soon be operating autonomously in complex environments filled with uncertainties and dynamic interactions. Many applications have already emerged as a result of these potential advances. A few examples are precision agriculture, space exploration, and search-and-rescue operations.

Most of the robotics applications involve a "search" component. In a search mission, the searcher is looking for a mobile target while the target is avoiding capture intentionally or obliviously. Some examples are environmental monitoring for population control and behavioral study of animal species, and searching for victims of a catastrophic event such as an earthquake.

In order to design search strategies with provable performance guarantees, researchers have been focusing on two common motion models. The first one is the *adversarial target* model in which the target uses best possible strategy to avoid capture. The problem is then mathematically formulated as a *pursuit-evasion game* where the searcher is called the "pursuer" and the target is referred to as the "evader". In pursuit-evasion games, when a pursuit strategy exists, it guarantees capture against any possible target strategy and, for this reason, can be seen as the worst-case scenario. Considering the worst-case behavior can be too conservative in many practical situations where the target may not be an adversary. The second approach deals with non-adversarial targets by modeling the target's motion as a stochastic process. In this case, the problem is referred to as *one-sided probabilistic search* for a mobile target, where the target cannot observe the searcher and does not actively evade detection. In this dissertation, we study both adversarial and probabilistic search problems. In this regard, the dissertation is divided into two main parts.

In the first part, we focus on pursuit-evasion games, i.e., when the target is adversarial. We provide capture strategies that guarantee capture in finite time against any possible escape strategy. Our contributions are mainly in two areas whether the players have full knowledge of each other's location or not. First, we show that when the pursuer has line-of-sight vision, i.e., when the pursuer sees the evader only when there are no obstacles in the between them, it can guarantee capture in *monotone* polygons. Here, the pursuer must first ensure that it “finds” the evader when it is invisible by establishing line-of-sight visibility, and then it must guarantee capture by getting close to the evader within its capture distance. In our second set of results, we focus on pursuit-evasion games on the surface of polyhedrons assuming that the pursuers are aware of the location of the evader at all times and their goal is to get within the capture distance of the evader.

In the second part, we study search strategies for finding a random walking target. We investigate the search problem on linear graphs and also 2-D grids. Our goal here is to design strategies that maximize the detection probability subject to constraints on the time and energy, which is available to the searcher. We then provide field experiments to demonstrate the applicability of our proposed strategies in an environmental monitoring project where the goal is to find invasive common carp in Minnesota lakes using autonomous surface/ground vehicles.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Technological developments has brought us advanced robotic platforms with much lower prices than ever in the past. As a result, we are witnessing emergence of enormous robotic applications not only in isolated manufacturing environments but also in our daily life, almost in every aspect: Housekeeping robots such as the vacuum cleaner robot Roomba [2]; medical robots such as da Vinci [3] in surgical operations and magnetic microbots [4] used to fight cancer cells or deliver medicine in blood vessels; driverless cars [5] used to prepare rich maps; precision agriculture aimed at enhancing crop management; and space missions such as Curiosity rover to explore unknown planets.

For most of the above robotic systems to work autonomously, technology is not enough: We also need sophisticated *motion planning* algorithms to *optimally* decide *where* should the robot move next such that it achieves its goal. A common goal in many robotic applications is *searching* for a mobile target. In a search mission, the searcher is looking for a mobile target while the target is avoiding capture intentionally or obliviously. Some examples of search missions are environmental monitoring, security and surveillance, and search-and-rescue to find victims of a catastrophic event such as an earthquake. The motion planning task here is to design a search strategy subject to the present limitations such that the target can be found with a given performance guarantee such as capture time or capture probability.

Many real-life applications of the search problem take place in complex environments and in the presence of sensing limitations. In addition, in most of the search applications, the target's trajectory is unknown. Therefore, in order to precisely define a search problem, one must specify the appropriate model for each of the following components: "target's motion model", "environment's complexity" and "sensing limitations". Fig. 1.1 shows an illustration of the different modalities of the search problem which we will explain shortly. In this dissertation, we solve different versions of the problem each of which is focused on one aspect of the search problem.

Target's Motion Model: In order to design search strategies with provable performance guarantees, researchers have been focusing on two common motion models. The first one is the *adversarial target* model in which the target uses best possible strategy to avoid capture against any search startegy. Here, the problem is in fact a non-cooperative game between the searcher and the target since they are pursuing conflicting goals: The searcher aims at "finding" the target while the target is "escaping" capture actively. The problem is then mathematically formulated as a *pursuit-evasion game* where the searcher is called the "pursuer" and the target is referred to as the "evader". In pursuit-evasion games, when a pursuit strategy exists, it guarantees capture against any possible target strategy and, for this reason, can be seen as the worst-case scenario. The worst-case capture guarantee is essential in many practical applications. For example, in sensitive applications such as missile tracking, the desire is to hit the missile (capture the target) with certainty.

Considering the worst-case behavior can be too conservative in many practical situations where the target may not be an adversary. The second approach deals with non-adversarial targets by modeling the target's motion as a stochastic process. In this case, the problem is referred to as *one-sided probabilistic search* for a mobile target, where the target cannot observe the searcher and does not actively evade detection. Formulating the problem as a probabilistic search problem, requires the knowledge of the target's movement model as a stochastic process. In addition, the capture guarantees are mostly probabilistic, e.g., capture probability.

Finally, we note that pursuit-evasion games are usually more complicated than probabilistic search problems since an adversarial target chooses the best escape strategy among infinitely many possible strategies as opposed to a stochastic target which is moving according to a single probability distribution.

Environment's Complexity: Another key factor in designing search strategies is the characteristic of the environment that the search mission is taking place in, which itself is contingent on the requirements of the corresponding application. For example, when the search is happening inside a building with capture condition as collocation in the same room, it might be enough to model the environment as a graph where capture occurs on its nodes. Other models to abstract the environment include: a plane with no boundary, a bounded region of a plane such as a polygonal area, a geodesic terrain modeled as the surface of a polyhedron, a bounded subset of the three-dimensional space or even more complex sub-spaces of n-dimensional manifolds. In addition, in all of these instances obstacles can be considered as well.

Although some of the aforementioned environments such as graphs and polygons have been studied well, very basic questions are open for non-planar environments. Even in the simple case of graph or polygonal setups, some variants of the problem in the presence of sensing limitation have remained unanswered for decades despite significant research focus on them.

Sensing Limitations: In addition to the environment's complexity, sensing limitations of the searcher for observing the location of the evader introduce another level of complication to the problem. In particular, the searcher's information about the location of the evader can be complete or partial. For example, a large network of cheap sensors, that are connected together, can provide the searcher with the complete information about the target's position. On the other hand, the searcher might be equipped with a single camera. Thus, it can see the target only when its line-of-sight is obstacle-free. As a result, in terms of sensing limitations, we can classify the search problem into two categories: complete sensing (full-visibility) and partial sensing (limited-visibility). In both categories, practically relevant versions remain unsolved.

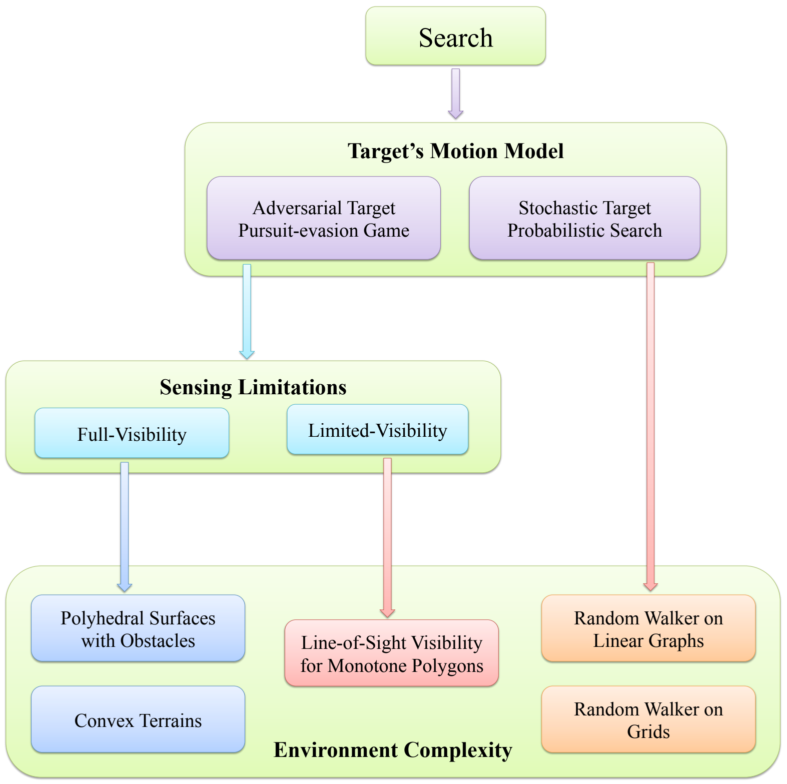


Figure 1.1: Different modalities of the search problem are shown: 1) target's motion model, 2) sensing limitations, and 3) environment's complexity. In the third box (environment's complexity) we have highlighted the specific variants that we study in this dissertation: the adversarial search problem with full-visibility on polyhedral surfaces and convex height-maps, the adversarial search problem with line-of-sight visibility in monotone polygons, and the probabilistic search problem for finding a random walker in a linear graph and also two-dimensional grid.

In this dissertation, we study both the adversarial and the stochastic target motion models. We study the adversarial search problem with full-visibility on polyhedral surfaces (with or without boundary) and also on convex height-maps (with boundary), the adversarial search problem with limited-visibility in monotone polygons, and the probabilistic search problem for finding a random walker in linear graphs and also two-dimensional grids subject to constraints on the available time.

The dissertation is organized as follows. In the following subsections of this chapter, we present a summary of our contributions. In Chapter 2, we provide an overview of related research. Chapter 3 presents the technical background that we will use throughout the dissertation. Chapters 4 and 5 are dedicated to pursuit-evasion games for capturing an adversarial target. In particular, in Chapter 4 we study a pursuit-evasion game with line-of-sight visibility (limited-visibility), and in Chapter 5 we study the game on three-dimensional polyhedral surfaces when the players have full knowledge of one another's location. In Chapter 6, we study the problem of finding a simple random walker (probabilistic search). Concluding remarks are presented in Chapter 7 where we also discuss some open problems and future research directions.

We now present an overview of our contributions in this dissertation. We first go over our result regarding adversarial target motion model. We then turn attention to our result on finding a discrete random walking target.

1.1 Contributions: Adversarial Target

In this section, we present a summary of our contributions in the domain of pursuit-evasion games (adversarial target). Here, the pursuer captures the evader if the distance between the evader and at least one pursuer is less than or equal to the capture distance. Later in Chapter 4 and Chapter 5, we will explore these contributions in full detail.

1.1.1 Partial Knowledge of Target's Location (Limited-Visibility)

A general question regarding pursuit-evasion games is the class of environments in which a single pursuer can capture the evader when the pursuer has line-of-sight vision. That is, the pursuer can see the evader only if the line segment connecting them is free of obstacles. We show that in *monotone polygons*, a single pursuer with line-of-sight visibility is enough to capture the evader. Our result provides a step toward characterizing the class of single-pursuer-win environments by showing that it includes monotone polygons.

With the limited vision power, the pursuer has to first find the evader when it disappears and then move toward the evader to capture it. We present a pursuit strategy which successfully combines search and capture. We are not aware of any other results which combine these two objectives for a single pursuer while providing guarantees about the outcome of the game. Our proposed capture strategy along with its analysis is provided in Chapter 4.

1.1.2 Complete Knowledge of Target's Location (Full-Visibility)

Many practical applications of pursuit-evasion games take place in non-planar environments; for example, when the searcher and the target are moving on the surface of a geodesic terrain; or when planning must be performed in the configuration space of a robotic manipulator. Despite the importance of the game in non-planar environment, little is known about its properties in such environments. In this dissertation, we study the game when it is played on the surface of a polyhedron. We show that on general polyhedral surfaces three pursuers suffice to capture the evader in finite time. Moreover, we show that our capture strategy works if the surface has "obstacles" i.e. subsets of the surface that neither player can enter. In other words, the surface is homeomorphic to a planar disk with a set of obstacles. A practical implication of our result is that three pursuers guarantee capture on terrains which is a special case characterized by unique height values for points in the two-dimensional plane (Fig. 1.2).

We then study the game in a more restricted sub-class of polyhedral surfaces: on the surface of a height-map (terrain). A terrain is obtained by assigning a single height value to each point in a bounded region of a plane in $\mathbb{R}^2$ (Fig. 1.2). We show that when the terrain is convex, a single pursuer with full-visibility captures the evader in finite time.

We will present our proposed capture strategies as well as their analysis in Chapter 5.

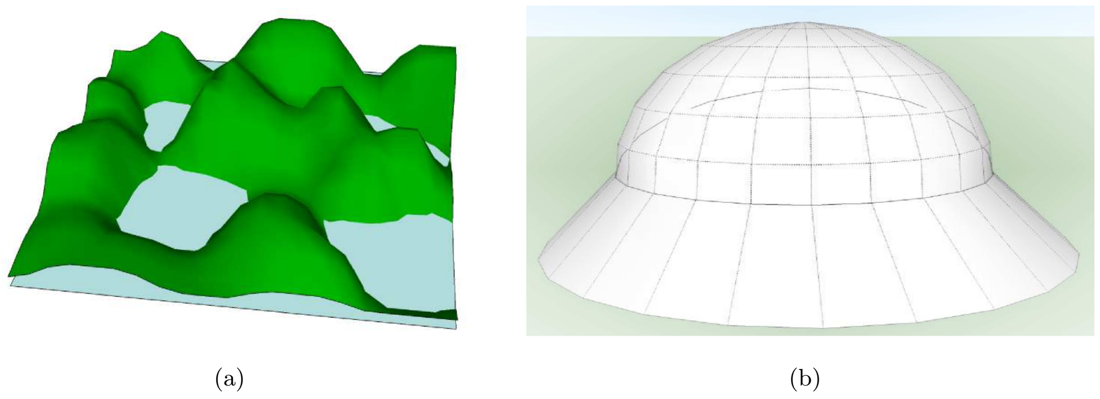


Figure 1.2: Two examples of a geodesic terrain (height-map), which is modeled as a polyhedral surface, is shown. (a) When the players are restricted to move outside water, the lakes are modeled as obstacles. (b) Here, the surface is composed of a half-sphere mounted on top of a cropped cone.

1.2 Contributions: Stochastic Target

We now discuss our results for the problem of finding a stochastic target (i.e., probabilistic search). Here, the problem is to find a target which is moving according to a simple random walk subject to time and energy restrictions such that the probability of detecting the target is maximized. We focus our attention on the problem when the searcher and the target are moving in a linear graph or a two-dimensional grid. The details of our results are presented in Chapter 6.

1.2.1 Simple Random Walker on Linear Graphs:

We first investigate the search problem when the target is a discrete one-dimensional random walker which moves in a linear graph. The target moves to neighboring nodes, i.e. to the `left` or `right`, with a given probability (Fig. 6.1(a)). The searcher on the other hand can choose to stay on its current node, or move to the right or to the left. Surprisingly, despite the simplicity of the problem at the first look, the problem is open.

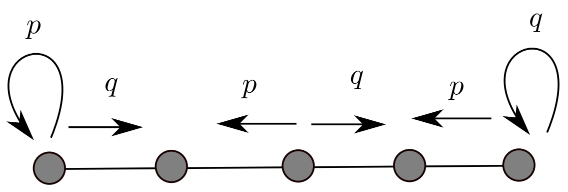


Figure 1.3: Target's motion model is a simple random walk: with probability p it moves one step to the left, and with probability $q = 1-p$ it moves one step to the right.

We study the problem under two detection criteria that we refer to as the *no-crossing* and *crossing* conditions. In the no-crossing model, the searcher detects the target if they are on the same node or if they take the same edge at the same time. In the crossing model, detection happens only if they land on the same node at the same time. For the no-crossing model an analytical solution for finding the optimal search strategy subject to energy and time constraints (where different costs are associated to move and stay actions) was presented in [6]. In this dissertation, we focus on the crossing model where we formulate the problem of computing the optimal search strategy as finding the optimal solution of a Partially Observable Markov Decision Process (POMDP) and also a Mixed Observability Markov Decision Process (MOMDP). We show that the solutions exhibit an interesting structure. Using this structure we focus on a set of strategies which we call the *uniform* strategies characterized by groups of right actions interleaved with stay actions, i.e. $(R^kS)^m$. We derive the best strategy in this class and show that $(R^2S)^m$ is performing close to the strategies found by the MDP methods. Finally, we provide preliminary experimental results to show that our model is useful in an environmental monitoring application where the goal is to search for invasive carp.

1.2.2 Simple Random Walker on Two-Dimensional Grids:

In addition to linear graphs, we also study the problem of finding a random walking target which is moving on an $N \times N$ grid. The target moves to neighboring nodes, i.e., to the `left`, `right`, `up` or `down` or chooses to `stay` on its current node with equal probability. We study a natural and easy-to-implement class of strategies we call *sweeping strategies*. In a sweeping strategy, the searcher picks a column and sweeps it entirely before choosing another column. We formulate the problem of computing the optimal sweeping strategy as the problem of computing the optimal solution of a Mixed Observability Markov Decision Process (MOMDP). We compare the MOMDP strategies against other heuristic strategies such as picking a column at random or picking every k^{th} column. We show that these strategies are performing fairly close to each other in terms of capture probability.

The rest of the dissertation is organized as follows. We cover the related literature in Chapter 2. The necessary technical background and the main tools are provided in Chapter 3. In Chapter 4 we study a pursuit-evasion game with partial knowledge of evader's location (line-of-sight visibility). In Chapter 5 we present our results for pursuit-evasion games with complete knowledge of evader's location on three-dimensional surfaces. We then study probabilistic search problems for finding a discrete random walking target in Chapter 6. Finally, we present concluding remarks and some open problems in Chapter 7.

The results presented in this thesis appear in [1, 6–16] and were funded by the National Science Foundation (grant numbers #0916209, #0917676 and #1111638).

Chapter 2

Related Work

In this chapter, we overview the existing literature on search and pursuit-evasion problems. We classify the literature into two categories based on the target's motion model (adversarial versus stochastic). First, in Section 2.1 we survey the results for capturing an adversarial target in graphs, planar environments and non-planar environments. We also give an overview of the related work for pursuit-evasion games with limited sensing. Then, in Section 2.2 we focus on random walks and present existing results on finding a stochastic target which is moving according to a Markovian model.

2.1 Adversarial Target (Pursuit-Evasion Games)

In this section, we first present an overview of the related research on pursuit-evasion games played in graphs. We then turn our attention to planar environments followed by the results in non-planar environments. We conclude the section with the related work regarding limited-visibility pursuit-evasion.

2.1.1 Cops and Robber Game in Graphs

In the game of cops and robber, a team of searchers (cops) are concerned with finding a mobile target (robber) hiding in a graph. The players move between adjacent nodes along the edges in the graph. The cops capture the robber if some of them can be co-located with the robber on the same node.

Various aspects of the problem such as players' relative speed [17], sensing capabilities [12, 18–20], and notion of capture [21, 22] have inspired a significant body of research. An overview of these results can be found in the survey paper by Chung et al. [23] and by Robin and Lacroix [24]. Nowakowski and Winkler [25], and independently Quillot [26], provided a characterization of cop-win graphs when the players can observe each other's location at all times. Megiddo et. al [27] provided a structural characterization of those graphs with cop-number less than k for $k \leq 1, 2, 3$. Aigner and Fromme [28] showed that three cops are sufficient for capture on planar graphs. When the initial locations of the players are given, Goldstein and Reingold [29] showed that the problem of determining whether k cops can capture the robber on a given undirected graph is EXPTIME-complete. Recently Kinnersley [30] showed that determining the cop-number (the minimum number of cops needed to capture a robber on a graph G) is EXPTIME-complete in general, even if the initial locations of the players are not specified.

In all the results above, the cops have "global visibility" i.e. they are aware of the location of the robber at all times. On the other hand, when the players cannot observe each other unless they are on the same node, the game is known as the hunter and rabbit game. Adler et al. showed that the hunter can capture the rabbit in $O(n \log n)$ time in a graph with n vertices [18]. Isler and Karnad [19] show that when each player can see the opponent if their distance is less than k (symmetric visibility powers), the cop can locate itself at distance at most k from the robber in any graph. In [31] the expected times required for capturing an adversarial robber and a drunk one (performing a random walk) is compared. Upper and lower bounds are derived for the ratio between these two values when the search is done on special graph structures.

2.1.2 Lion and Man Game in Planar Environments

In the previous sub-section, the cops and robber game takes place in graphs. Many robotics applications are naturally formulated as geometric versions of the cops and robber game where the players move in a continuous space instead of a graph. A classical geometric pursuit-evasion game is the *lion and man game.* In this game, the lion pursues the man, and the man tries to escape capture.

The lineage of the lion and man game traces back to Rado's classic version from the

1930s [32] where the game takes place in a circular arena, and both the lion and the man have equal speed. The lion wins if it becomes collocated with the man in finite time. Intuitively, the lion (pursuer) should win the game: if it moves directly toward the man, the man has to step back in the same direction to maintain the separation between the players. Following this strategy, the man will eventually hit the boundary and thus he cannot escape forever. It turns out that the analysis of this simple "greedy" strategy is not straightforward since the turn angle can be arbitrarily small. In fact, when time is continuous, players move simultaneously and capture requires colocation, man can avoid capture indefinitely by following a gently spiraling path [32]. On the other hand, in the turn-based version where the players take turns, it has been shown that the lion captures the man by following the *lion's strategy* which was generally accepted in folklore [32]. In this strategy, the lion moves such that it is always on the radius between the man and the center. The capture time of this turn-based strategy is $O(R^2)$ where R is the radius of the environment. Sgall [33] uses a similar strategy to show finite time capture when the game takes place in the non-negative quadrant of the plane. Alonso et al. [34] proposed a more sophisticated strategy which guarantees capture in $O(R \log \frac{R}{r})$ steps where r is the capture radius.

Isler et al. [35] adapted lion's strategy for pursuit in a simply connected polygon P. First, the pursuer starts at point c in the polygon, which is typically a boundary vertex. Thereafter, the pursuer always moves onto the shortest path between c and the evader, getting as close to e as possible. For a polygon P with n vertices and diameter $diam(P) = \max_{u,v \in P} d(u, v)$, Isler et al. [35] proved that the capture time is $O(n \cdot diam(P)^2)$. Beveridge and Cai [36] proved that the capture time is $O(diam(P)^2)$. Zhou et al. [37] show that adding a second pursuer reduces capture time to $O(diam(P))$ in simply connected domains.

The game has been also studied in the presence of obstacles. Bhadauria et al. [38] show that three pursuers can capture the evader in any polygonal environment with obstacles. Zhou et al. [37] show that three lions are still enough for capture in the presence of obstacles even when the domain is not polygonal. Their strategy guarantees capture in time $O(h \cdot diam(P))$ where h is the number of obstacles. Bhattacharya and Hutchinson [39] study the game in planar environments containing obstacles when the pursuer's goal is to maintain visibility of the evader for the maximum possible time and the evader's goal is to escape the pursuer's sight as soon as possible. They present necessary and sufficient conditions for tracking as well as Nash equilibrium strategies for the players.

2.1.3 Lion and Man Game in Non-Planar Environments

The lion and man game has been studied in non-planar environments as well. Kopparty and Ravishankar [40] showed that in $\mathbb{R}^d$, $d+1$ lions can capture the man if and only if the man starts inside their convex hull. Bopardikar and Suri [41] study *k-capture* in m-dimensional Euclidean spaces where at least k pursuers must simultaneously reach the evader's location to capture it; in addition, if fewer than k pursuers reach the evader they will be destroyed by the evader. They show that the necessary and sufficient condition for capture is the existence of a time instance that the evader lies inside in the pursuers' k-Hull. Alexander et al. [42] study pursuit in CAT(0) environments where the curvature is non-positive, and show that a single pursuer with positive capture radius can eventually capture the evader by greedily moving toward it. Beveridge and Cai [36] show that in any CAT(0) environment one pursuer has a winning strategy under the collocation capture condition as well (i.e., zero capture distance).

The problem is also studied on the surface of a polyhedron which models the search applications on geodesic terrains. Klein and Suri [43] showed that four pursuers with zero capture distance are sufficient to capture the evader on a polyhedral surface with genus zero. In this dissertation, we show that three pursuers are enough for capture when the capture distance is non-zero even in the presence of obstacles on the surface [11, 14] (Section 5.2). In [16] we show that the class of convex terrains, which includes positive curvature examples, are still single pursuer-win (Section 5.3).

2.1.4 Lion and Man Game with Limited-Visibility

The lion and man game has been also studied when the pursuer has only line-of-sight vision. That is, the pursuer can see the evader only if the line segment connecting them is free of obstacles. This variant models robotics applications where the pursuer is a robot equipped with a camera or a laser scanner. When the goal of the pursuer is to just find the evader the problem is called the visibility-based pursuit evasion problem [44].

The necessary and sufficient conditions for a simple polygon to be searchable by a pursuer with various degrees of visibility power is presented in [21]. Guibas et al. show that for an arbitrarily fast evader, the minimum number of pursuers required in the worst case is $\Theta(\log n)$ for simply-connected polygons and $\Theta(\sqrt{m} + \log n)$ for a polygon with m holes [44]. Klein and Suri [45] showed that in a polygon with n total vertices and h holes, $O(\sqrt{h} + \log n)$ lions with visibility can capture the man if the minimum distance between any two vertices is lower bounded by the step size. If the environment can have arbitrarily small "features" compared to the speed of the players, then the number of pursuers can be as high as $\Omega(n^{2/3})$ [46]. In Chapter 4 we study the game when the goal of the pursuer is to capture the evader in the sense that it gets within the capture distance of the evader. We show that a single pursuer can successfully capture the evader in *monotone* polygons [8, 12]. Berry et al. [47] present a line-of-sight pursuit strategy for capture in strictly sweepable polygons, which are a generalization of monotone polygons.

Bopardikar et al. [48] study the lion and man game in the plane where every player has identical sensing and motion ranges restricted to disks of given radii. In their model, the evader is reactive in the sense that it only moves when it sees the lion. The authors provide a lower bound condition on the ratio between the sensing radius and the step size such that the pursuer can trap the evader inside its sensing circle when the environment is convex [48]. Karnad and Isler [49] show that in the positive quadrant of the plane a pursuer with a bearing measurement sensor can reduce its distance to the evader to the step size in finite time.

We next discuss related results in the domain of probabilistic search problems.

2.2 Stochastic Target (Probabilistic Search)

We now focus on the related research for finding a random walker. We first overview the search problem for finding a stochastic target on graphs in Section 2.2.1. We then provide some known results related to random walks in Section 2.2.2.

2.2.1 Two-Cell and N-Cell Problems

In one-sided probabilistic search problems, the target cannot observe the searcher and does not actively evade detection. Instead, the target's motion is modeled as a stochastic process. The simplest setting is when the target moves in discrete time steps in a discrete world consisting of only two cells according to a Markovian motion model [50]. It turns out that even this simple variant, which is referred to as the *two-cell* problem, is challenging. Pollock [50] uses dynamic programming to derive search strategies that minimize the expected number of looks to detect the target, or maximize the detection probability within a given number of looks for special cases of the two-cell problem. Wilson [51] provide a necessary and sufficient condition on the initial distribution of the target's position such that a search plan with finite expected capture time exists. Dobbie [52] studies the continuous time motion model and solves for optimal strategies that minimize the expected time to detection, or maximize the probability of detection in a given time in the hope to derive formulations that are easier to generalize to more than two cells. Kan [53] also attempted to generalize the problem to N cells by characterizing optimal strategies for special cases, for example when the cells form a clique and the target moves between all cells with equal probability. In fact, it has been shown that the problem of detecting a target, stationary or mobile, in a grid world within a fixed time horizon is NP-complete [54]. As a result, approximation methods have also been studied to tackle the problem. In this regard, branch-and-bound methods [55, 56] and POMDP formulations [57, 58] are popular. For example, Lau et al. [59] use a branch-and-bound framework to find an upper bound on detection probability for relatively small environments with around 20 nodes. Hollinger et al. [57] formulate the problem of maximizing the expected probability of finding the target at the earliest possible time by multiple searchers as a POMDP and use the sub-modularity of the joint discounted reward function to provide a constant factor approximation sequential allocation algorithm.

In Chapter 6, we study the problem of finding a target which is moving according to a simple random walk on a linear graph. We next review some properties of random walks as well as some known results regarding the search problem.

2.2.2 Random Walks

In this section, we focus on the literature related to random walks. Random motions, both as discrete random walks and continuous diffusive motions, have been extensively studied as models of unknown animals motions or complex physical processes [60]. In particular they are widely used in the literature to simplify pursuit-evasion games and absorption (or search) processes. A large number of interesting properties closely related to searching missions are collected in [61] including: first passage probability (the probability for the random walker of visiting for the first time a given point at a given time), survival probability (the probability that the random walk has not been found at a given time) and mean capture time (the expected time to be found). Various characteristics of random walks in general graphs have been studied in [62]. Examples are hitting time, which is the expected number of steps before a node is visited, and cover time, which is the expected number of steps to visit every node at least once.

Although one-dimensional random walks might seem too simple, they present several interesting behaviors and properties and are still source of open problems. The survival probability of a particle that performs a random walk on a chain where traps are uniformly distributed with known concentration is studied in [63] and an asymptotically exact solution is provided. In [64], the authors study the survival probability of a prey on a line which is chased by more than one diffusive predator. The same problem but in a semi-infinite line where the boundary represents a haven for the prey is presented in [65]. None of these works have addressed the capture problem restricted to constraints on the energy of the system, or on the maximum time for the chase. This is the subject of our work in Chapter 6.

In the next chapter, we present the technical tools that we use to solve the pursuit-evasion games in Chapters 4 and 5 and also the probabilistic search problem for finding a random walker in Chapter 6.

Chapter 3

Technical Background

In this chapter, we present the basic techniques that are used in the following chapters. First, in Section 3.1 we discuss main ideas and sub-strategies for capturing adversarial targets in pursuit-evasion games, which are later exploited in Chapters 4 and 5. Then, in Section 3.2 we describe Partially Observable Markov Decision Processes (POMDP) and Mixed Observability Markov Decision Processes (MOMDP), which we exploit to design search strategies for finding a random walking target later in Chapter 6.

3.1 Pursuit-Evasion Games

Let us start by giving a formal description of pursuit-evasion games[1]. In a pursuit-evasion game, the pursuers' goal is to capture the evader while the evader is trying to avoid capture as much as possible. Throughout the dissertation we will interchangeably refer to the pursuer as the *lion*, and the evader as the *man*. The notion of capture that we consider is whether the distance between the evader and at least one pursuer is within a fixed *capture radius* denoted by r. We assume that the players have the same maximum step-size; we employ a unit step-size which means that the players can move along a path contained in the environment, of length at most one. The unit step size assumption is both standard and convenient. Finally, we will consider discrete time turn-based version of the game where the players take turns and each turn takes a unit time-step. We focus on the turn-based version of the game in order to avoid the

[1] The material in this section appears in [7].

following pathology: When time is continuous and the players move simultaneously, the evader can avoid capture indefinitely by following a gently spiraling path [32].

We continue this section by presenting the main ideas and the basic techniques in designing pursuit strategies. In particular, we first explain *lion's strategy*, and *rook's strategy*. Meanwhile, we describe main concepts such as *guarding*, and *making progress*. We then proceed with more advanced topics such as *projection mappings* for *guarding shortest paths*.

In order to capture the evader, a general pursuit strategy consists of two main phases. First, the evader is expelled from a subset of the environment, and this subset is protected thereafter. Second, the protected subset is gradually grown until the whole environment is cleared and the evader is captured. These two phases are referred to as **guarding** and **making progress** respectively.

As a demonstrative example, suppose that the turn-based game is played in a square region; the pursuer can observe the exact location of the evader, and the capture radius is $r = 0$. A first intuitive idea for guarding is to locate the pursuer on the line segment L between two arbitrary boundary endpoints, and prevent the evader from crossing it. If we can also push L towards the evader, then the progress goal would be achieved as well.

Here is our first example of a guarding strategy. The pursuer can guard L by positioning itself on the **vertical projection** of the evader onto L (Fig. 3.1(a)). The vertical projection has the property that it is closer to all the points along L than the evader itself. As a result, if the evader tries to cross L, the pursuer (which is on the evader's projection) will capture it during its next move. We make this argument more rigorous in Section 3.1.7.

Although the vertical projection idea succeeds in restricting the evader to one side of L for the rest of the game, the pursuer fails to achieve its second goal: shrinking the evader's region. Indeed, suppose that the evader takes a full unit step parallel to L, moving from e_1 to e_2 with $|e_1e_2| = 1$ (Fig. 3.1(b)). The evader's vertical projections onto L, denoted by p_1 and p_2 respectively, satisfy $|p_1p_2| = 1$. Therefore, the pursuer exhausts its movement budget just to keep up with the evader. Now, if the evader reverses its direction after each step, the pursuer is forced to move between the two fixed points p_1 and p_2. Consequently, the pursuer cannot move L towards the evader, and the evader can escape forever.

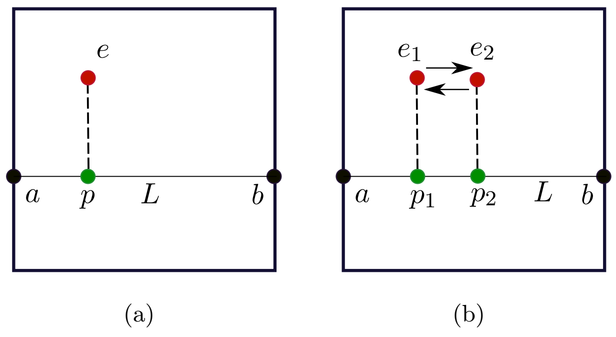


Figure 3.1: **(a)** The pursuer can prevent the evader from crossing L by positioning itself on the vertical projection of the evader. **(b)** Since e_1e_2 is parallel to L and $|e_1e_2| = 1$, we have $|p_1p_2| = 1$. Thus, the pursuer is stuck on L between the two points p_1 and p_2.

There are three approaches to tackle the issue above and achieve progress: (1) Lion's strategy, (2) Multiple guarding pursuers, and (3) Rook's strategy. In the following subsections, we will present an intuitive description of each approach using our square example. But before that, let us first present the notation that is used in this chapter.

3.1.1 Notation

The game environment is denoted by $\mathcal{S}$ with boundary $\partial\mathcal{S}$. We refer to the subset of $\mathcal{S}$ that the evader cannot enter without being captured as the **cleared** region. The remaining part of $\mathcal{S}$ is referred to as the **contaminated** region. We refer to a shortest path in $\mathcal{S}$ between points x and y by $\Pi(x, y)$. The shortest distance between x and y, i.e., the length of $\Pi(x, y)$, is denoted by $d(x, y)$. We denote $\max_{u,v\in\mathcal{S}} d(u, v)$ by $\mathrm{diam}(\mathcal{S})$. When we require that a distance is measured within a subset, such as to $Q \subseteq \mathcal{S}$, we write $d_Q(x, y)$. We use $B(x, r) = \{y \in \mathcal{S} \mid d(x, y) \leq r\}$ to denote the ball of radius r centered at x.

We are now ready to discuss the first approach called the *lion's strategy* which is known in folklore [32] since 1950s.

3.1.2 Approach 1. The Lion's Strategy

In the **lion's strategy**, the pursuer can simultaneously guard and make progress. In this strategy, the pursuer (lion) starts at an arbitrary center $p^0 = c$. In round $t \geq 1$, the pursuer makes a **lion's move**, meaning that it moves from p^{t-1} to the point $p^t \in B(p^{t-1}, 1)$ on the line segment connecting c to e^t such that p^t is closest to the evader (Fig. 3.2).

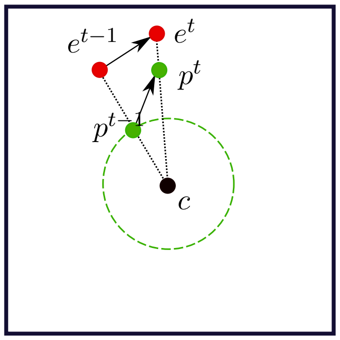


Figure 3.2: Lion's strategy in a square region.

Simple trigonometric arguments can be used to show that the lion's strategy captures the man. The proof exhibits the two essential ingredients for verifying that a pursuit strategy succeeds in finite time. First, we establish an **invariant** that the pursuer(s) maintain throughout the game. For lion's strategy, this invariant is that p was located on the radius between the center and the evader. Second, we need a **measure of progress** to show that the game ends in finite time. Let d and d' denote the distance between the center c and the lion before and after a move, respectively (Fig. 3.3(b)). Let α be the angle between ce_1 and p_1p_2. We have $d'^2 = (d + \cos\alpha)^2 + \sin^2\alpha = d^2 + 2d\cos\alpha + 1$ (Note that $p_1p_2 = 1$). We first show that there exists a point p_2 on ce_2 such that $\alpha \leq \frac{\pi}{2}$. This is because $e_1e_2 \leq 1$, and hence $p_1q \leq 1$ where q is a point on ce_2 such that qp_1 is prependicular to ce_1 (Fig. 3.3(a)). As a result of $\alpha \leq \frac{\pi}{2}$ we have $d'^2 \geq d^2 + 1$. When coupled with the invariant, this guarantees capture after at most R^2 rounds.

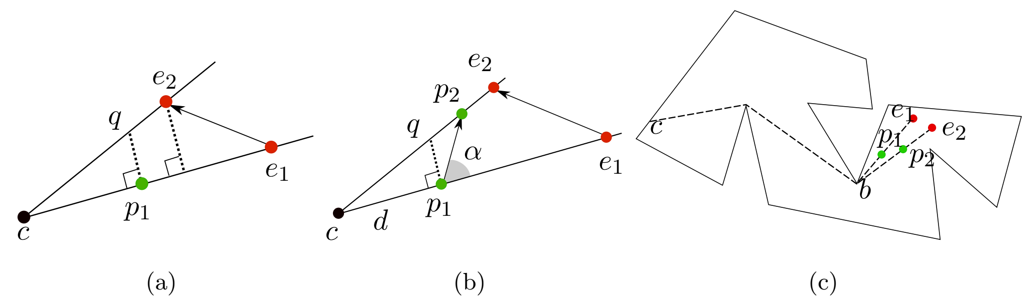


Figure 3.3: In lion's strategy, the pursuer can increase its distance from the center by at least $d' - d \geq \frac{1}{2R}$. This is because: **(a)** The length of p_1q is less than one. As a result, there exists a point p_2 on ce_2 which is closer to e_2 than q, and moreover is at distance one from p_1. **(b)** In fact, $\alpha < \frac{\pi}{2}$. **(c)** Lion's strategy in a polygon.

Isler et al. [35] adapted lion's strategy for pursuit in a simply connected polygon P. First, the pursuer starts at point c in the polygon, which is typically a boundary vertex. Thereafter, the pursuer always moves onto the shortest path between c and the evader, getting as close to e as possible. Note that this shortest path could interact with the boundary of the polygon, in which case it will be a piecewise linear path (Fig. 3.3(c)). The extended lion's strategy uses the same invariant (being on the shortest path between c and e) and the same measure of progress (increasing $d_\pi(c, p)$ at a constant rate) as lion's strategy. In Chapter 4, we extensively use lion's strategy in our proposed capture strategy.

3.1.3 Approach 2. Multiple Guards

In the second approach, an additional pursuer is added to guard another line segment. Suppose that we have two pursuers, each of which guards a line by positioning itself on the evader projection. The first pursuer p_1 guards a horizontal line segment L_1, while the second pursuer p_2 guards the vertical line segment L_2 (Fig. 3.4(a)). This traps the evader in one quadrant of the square. If the evader moves horizontally, then p_1 mimics this move while p_2 makes one unit of progress by advancing its guarded line. Likewise, if the evader moves vertically, then p_2 keeps pace while p_1 advances its guarded line. More generally, the Pythagorean theorem shows that at least one of the pursuers can advance its guarded line by $\sqrt{2}/2$ units in each turn. This means that the evader will be cornered by the pursuers after a finite number of moves, after which one of the pursuers has a capture move.

In Chapter 5, we will combine the idea of multiple guarding pursuers with path projection mapping, which we will discuss shortly in Sub-section 3.1.7, to show capture on polyhedral surfaces.

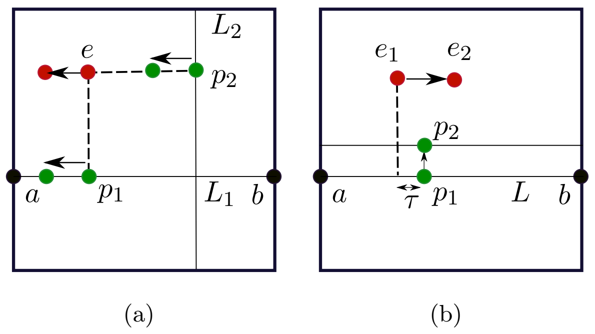


Figure 3.4: **(a)** Two pursuers are guarding L_1, L_2 by staying on the projection of the evader. One of them can make progress after each move. **(b)** A single pursuer can push L towards the evader if it stays behind the projection of the evader.

3.1.4 Approach 3. The Rook's Strategy

In both of the approaches above, capture is guaranteed for zero capture radius. Pursuit games with capture radius $r > 0$ are also common in the literature. The third approach takes advantage of the non-zero capture radius to overcome the stalemate in Fig. 3.1(b). In Chapter 5, we will leverage the rook's strategy in our proposed strategy to capture the man on the surface of a convex terrain (convex height-map). To build intuition, consider a chess endgame for a white rook and white king versus a solitary black king.

The **rook's strategy** works as follows: One row at a time, the white rook reduces the area available to the black king. This is achieved with support of the white king who trails the projection of the black king onto the row guarded by the rook. Specifically, suppose the white rook is at row i, column 1, which guards row i. The black king is at row $i' > i$ and column j, and the white king is at row $i - 1$, column $j - 1$. From this position, the white king trails the black king until the column separation of the white rook and white king is at least two (Fig. 3.5(a)). After achieving this separation, the white player can make progress as follows. If the black king moves back on to column $j-1$, then white can move the rook to row $i+1$ and push the black king further up (Fig. 3.5(b))[2]. If the black king moves away from the white king, then the white king keeps trailing him. Eventually, the black king is trapped in a single row, where white can checkmate.

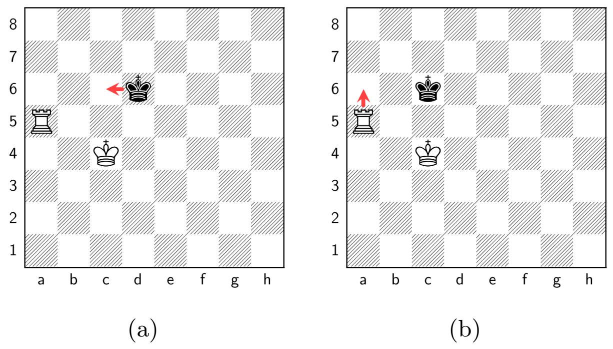


Figure 3.5: The rook-and-king chess endgame is shown in (a) and (b). **(a)** The white rook is one row below the black king. The white king is one row below and (at least) two columns to the right of the white rook, and is just to the left of the black king. **(b)** After the black king moves leftward, the white rook makes progress.

In rook's strategy, the pursuer combines the roles of the white rook and white king (because the evader cannot threaten the pursuer). Like the white rook, the pursuer guards a horizontal frontier line. Like the white king, the pursuer remains offset from the projection of the evader. Whenever the evader moves horizontally "back" toward the projection, the pursuer can make progress by vertically advancing the frontier line. Eventually, the evader will be squeezed between the guarded path and the boundary, where it will be caught. The rook's strategy is an alternative to lion's strategy for *actively* chasing the evader. Simply put, rook's strategy is designed to *simultaneously* guard and make progress.

Let us formally present the rook's strategy by adapting the chess strategy introduced

[2] This is because the black king cannot move to row i since the reachable cells are being guarded by the white king. Furthermore, in the extreme case where $i' = i+1$, the black king cannot stay in row i' because of the two column separation between the white king and the white rook.

above to the lion and man game in a square environment. Suppose that the pursuer is guarding a straight line segment Π between two boundary points. As we observed earlier in this chapter (Fig. 3.1(b)), when $r = 0$, guarding Π requires $p = \pi(e)$. As a counter-strategy, the evader can oscillate horizontally between two points, unit distance apart (Fig. 3.1(b)). In response, the pursuer must exhaust its movement budget just to keep pace, so the guarded path never advances. However, when we have a positive capture radius $r > 0$, the pursuer guards Π as long as $0 \leq d(p, \pi(e)) < r$. We claim that this slack allows the pursuer to reliably advance the guarded path.

From here forward, we consider the game with positive capture radius $r > 0$ and fix a constant $0 < \tau \leq \min\{r, 1/2\}$. Given a projection π onto path Π, we say that p is in **rook position** when $0 \leq d(p, \pi(e)) \leq \tau$. Suppose that the pursuer is offset τ units to the right of the evader projection (Fig. 3.6(a)). If the evader moves leftwards one unit, then the pursuer matches pace and maintains this offset. However, if the evader moves rightward, then the pursuer switches to using a leftward offset instead (Fig. 3.6(b) and Fig. 3.6(c)). As a result, the pursuer only needs to move rightward $1 - 2\tau$ units, which means that it can move diagonally to achieve at least $(1 - (1 - 2\tau)^2)^{1/2} > \tau$ units of upward progress. The key observation is that the evader must move rightward at some point, since $\text{diam}(\mathcal{S})$ is finite, so the pursuer makes at least τ units of vertical progress every $\text{diam}(\mathcal{S})$ steps.

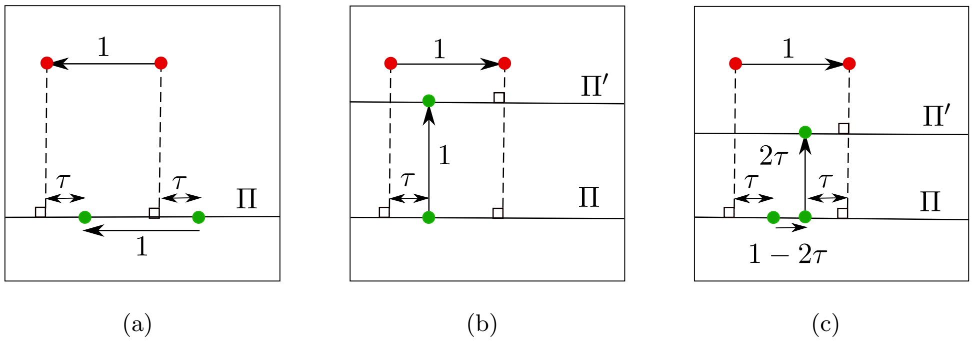


Figure 3.6: Illustration of the rook strategy when $\tau < 1/2$ is shown. **(a)** If the evader moves to the left, the pursuer moves in the same direction for one unit. **(b)** If the evader moves to the right, the pursuer pushes Π to Π'. When $1 - \tau \leq r$, the distance between Π and Π' is one unit. **(c)** When $1 - \tau > r$, the distance between Π and Π' is 2τ. Here notice that since $-\tau \geq -r$ and $1 - \tau > r$ we have $1 - 2\tau > 0$.

We next compare the rook's strategy with the lion's strategy.

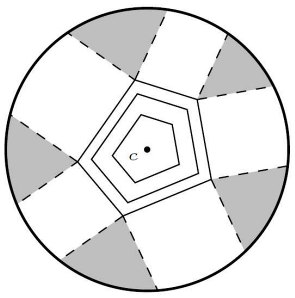
3.1.5 Lion's Strategy Versus Rook's Strategy

It is worth comparing how lion's strategy and rook's strategy guard and attack simultaneously. During lion's strategy, the pursuer consistently radiates outwards from its initial point c. Let p^t be the location of the pursuer at time t and let $B^t = B(c, d^t)$ where $d^t = d(c, p^t)$. Each pursuer move decreases the evader territory: after time t, the evader cannot step into the region B^t. Indeed, the pursuer is actually located on the closest point projection onto ∂B^t, and it is useful to view p as guarding the expanding sequence of **wavefronts** $\partial B^1, \partial B^2, \ldots, \partial B^t$. The rook's strategy also controls a sequence of advancing wavefronts. One advantage of rook's strategy is that its wavefronts are straight lines (or piecewise linear paths). This makes rook's strategy a natural fit for polygonal and polyhedral environments, where it can lead to simpler pursuit algorithms than those employing lion's strategy.

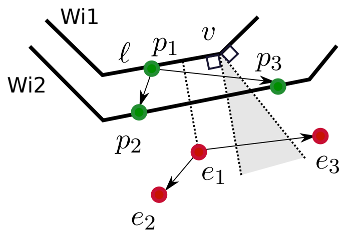
3.1.6 Centered Rook's Strategy

It may seem that rook's strategy is only suited for rectilinear pursuit, since it is crucial that the evader must "turn around" when it encounters the left or right boundary. We can make rook's strategy more powerful by expanding the wavefronts from a central point similar to the lion's strategy. This **centered rook's strategy** will guard wavefronts that bound a family of convex polygons (rather than regions with curved boundaries).

Consider a pursuit-evasion game in a convex polygon $\mathcal{S}$ with capture radius $r > 0$. Fix an offset $0 < \tau \leq \min\{r, 1/2\}$. Pick a center $c \in \mathcal{S}$ and let A be a convex polygon such that $\max_{x \in A} d(c, x) = 1$. We explain how p can guard a monotonically increasing family of regions $A_i = \{b_i x \mid x \in A \cap \mathcal{S}\}$ where $b_1 = 1$ and $b_{i+1} \geq b_i$. We call b_i the **radius** of the guarded region. We use the closest point projection (which returns consistent values, even when the evader circumnavigates A_i). Note that this projection onto A_i partitions $\mathcal{S}$ into distinct areas, according to the pre-images of the vertices and sides of A_i, see Fig. 3.7(a).



(a)



(b)

Figure 3.7: The centered rook strategy. (a) The expanding wavefronts, and the pre-images of the edges and vertices of polygon A_t centered at c. (b) Two progress events are depicted. Suppose that the evader is currently at e_1 and the pursuer is guarding W_1 on the edge ℓ. If the evader's projection stays on ℓ the pursuer moves to p_2 by rook's strategy. If the evader crosses the pre-image of v, the pursuer makes progress by moving to p_3.

The pursuer starts at c. On its first move, p moves to within τ of the closest point projection on A, so that it is now guarding $A_1 = A$. Now, suppose that p currently guards A_t. We claim that in finite time, the pursuer can increase the radius b_t of the guarded region by a constant amount. Suppose that the evader projection $\pi(e)$ is on side ℓ of A_t, so the evader is in the pre-image of ℓ. See Fig. 3.7(b). If the evader never leaves this pre-image, then the pursuer makes progress as in regular rook's strategy. Otherwise, the evader must step into (or through) the pre-image of a vertex of A_t. As the evader crosses the pre-image of vertex v, its projection remains fixed on v. This frees up part of the pursuer's movement budget for progressing to the next wavefront (Fig. 3.7(b)). For full details, see Chapter 5, Section 5.3.

We next present a more advanced technique called **projection mapping**, which is used to guard shortest paths. Later in Chapter 5, we will see an application of shortest path guarding in capturing the evader on the surface of a polyhedron.

3.1.7 Path Guarding and Projection Mappings

In this section, we describe how a shortest path can be guarded by a pursuer. This capability turns out to be a powerful subroutine for solving the lion and man game. We will use this idea in Chapter 5 to show capture on the surface of a polyhedron even in the presence of obstacles. We start with a formal definition of guarding. Then, we show how to use projection mappings to guard subregions.

Definition 1 (Guarding)**.** *A pursuer p* guards *a subregion Q when p can immediately respond with a capture move whenever the evader e steps into or through Q.*

Fig. 3.1(a) shows a guarding example, and also includes our first example of a projection. Let $Q = \Pi$ be a horizontal line segment connecting two boundary points. For any point $e \in \mathcal{S}$, let $\pi(e)$ be the **vertical projection** of e onto Π. Basic geometry shows that if e moves to a point e' then $d(\pi(e), \pi(e')) \leq d(e, e')$. This means that if $p = \pi(e)$ then it can move to $\pi(e')$, so the pursuer can maintain this property indefinitely. Furthermore, if e steps onto or across Π then p can respond with capture. In summary, a pursuer positioned at $\pi(e)$ can guard Π, restricting the evader to a subset of the environment. Moreover, it is clear that a pursuer can achieve $p = \pi(e)$ in finite time: start at the left endpoint of Π and then walk rightward at full speed until reaching $\pi(e)$.

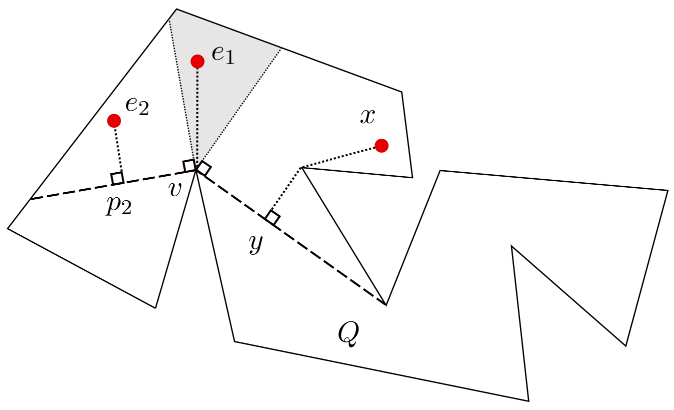


Figure 3.8: The closest point projection is illustrated in a simply connected polygon.

With this intuitive vertical projection in mind, we are now ready to consider some more flexible projection mappings. We start with a general definition of a projection. Simply put, the mapping $\pi : \mathcal{S} \to Q$ is a projection provided that it is the identity map on Q, and that the mapped points are as close or closer together as the original points.

Definition 2 (Projection)**.** *A projection $\pi : \mathcal{S} \to Q$ is a function such that (1) if $x \in Q$ then $\pi(x) = x$, and (2) for all $x, y \in \mathcal{S}$, we have $d_Q(\pi(x), \pi(y)) \leq d_{\mathcal{S}}(x, y)$.*

Suppose that the pursuer is located at $p = \pi(e) \in Q$ where $\pi : \mathcal{S} \to Q$ is a projection. The previous argument for a vertical projection generalizes: the pursuer can guard Q [38]. We describe two useful projections that are well-suited for more complicated environments.

3.1.7.1 Closest Point Projections

Let $\mathcal{S}$ be a simply connected polygon. Given boundary points $u, v \in \partial\mathcal{S}$, let Π be the unique shortest path between them. Define the *closest point projection* $\rho : \mathcal{S} \to \Pi$ to be the mapping that takes $x \in \mathcal{S}$ to the point $y \in \Pi$ that is closest to x, see Fig. 3.8. The vertical projection above is just a special case of the closest point projection. As in the square region, a pursuer can establish a guarding position on Π and maintain it thereafter, trapping the evader in a sub-polygon.

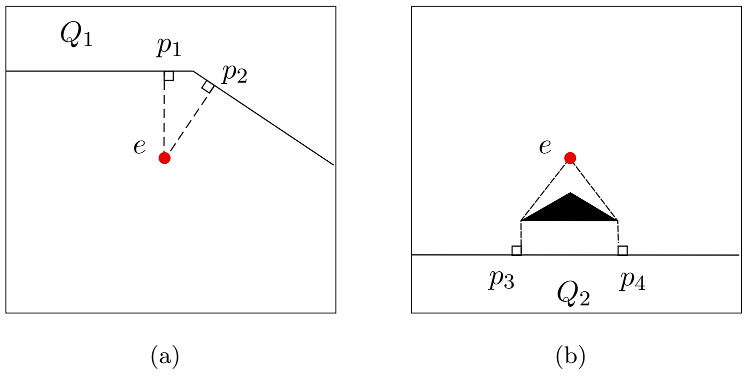


Figure 3.9: Two examples where the closest point mapping is not a projection mapping. In these examples, the closest point is not unique. **(a)** In the first example, since Q_1 is not convex, there are two points that are closest to e in Q_1 (p_1 and p_2). **(b)** In the second example, due to the obstacle in $\mathcal{S} \setminus Q_2$, the point e has two closest points in Q_2 (p_3 and p_4). Therefore, the evader can move such that its closest point moves faster. Thus, the pursuer cannot stay on the closest point of the evader.

3.1.7.2 Path Projections

The situation changes once we introduce obstacles to the environment: the closest point mapping becomes ill-defined when there are obstacles between e and Π, see Fig. 3.9. Bhadauria et al. [38] introduced an alternate type of projection that is less intuitive, yet robust in the presence of obstacles. Let $a, b \in \partial\mathcal{S}$ and let Π be a shortest (a, b)-path, which we consider to be *anchored* at a. The *path projection* of the point $x \in \mathcal{S}$ is the point $y \in \Pi$ such that $d(a, y) = d(a, x)$. If $d(a, x) > d(a, b)$, then we simply define $\pi(x) = b$. See Fig. 3.10 for an example. The path projection remains unique, even when there are obstacles in the environment. A pursuer on the path projection $\pi(e)$ can guard Π, meaning that $d(\pi(e), \pi(e')) \leq d(e, e')$, and that the pursuer can capture the evader whenever it crosses Π [38].

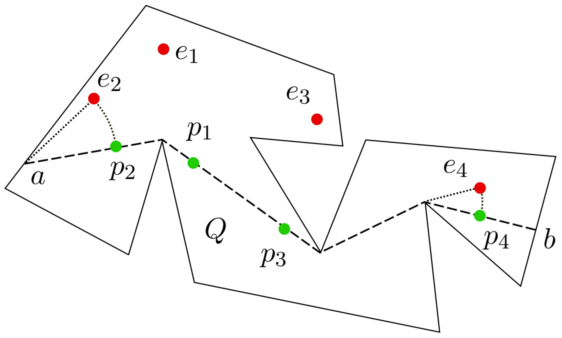


Figure 3.10: The path projection anchored at a.

Verifying that a path projection satisfies $d(\pi(x), \pi(y)) \leq d(x, y)$ is straight-forward, though there are multiple cases to consider depending on the distance of $x, y \in \mathcal{S}$ from the anchor a. The validity of this projection means that a single pursuer can guard a path and thereby restrict the sub-environment available to the evader, even in the presence of obstacles.

So far we have discussed important concepts for capturing an adversarial evader used in pursuit-evasion games. We will leverage these techniques in Chapters 4 and 5 where we study different versions of the lion and man game. Let us next discuss the tools for solving probabilistic search problems: POMDPs and MOMDPs.

3.2 Probabilistic Search

In this section, we present a description of Partially Observable Markov Processes (POMDPs). We also discuss Mixed Observability Markov Decision Processes (MOMDPs), which are special classes of POMDPs developed with the purpose of dealing with the curse of dimensionality in POMDPs. Later in Chapter 6, we use these techniques to design search strategies for finding a random walker.

3.2.1 Partially Observable Markov Decision Process (POMDP)

We start by a brief overview of Markov Decision Processes (MDPs) [66]. An MDP is described by a tuple $(\mathbb{S}, \mathbb{A}, \mathbb{T}, \mathbb{R})$ where $\mathbb{S}$ is the set of possible states, $\mathbb{A}$ is the set of actions, $\mathbb{T}$ is the probability of transitioning between the states as a result of performing each action, and $\mathbb{R}$ is the reward collected for each transition. Here, $\mathbb{T}$ and $\mathbb{R}$ are represented as matrices: The entries of the transition probability matrix $\mathbb{T}(s_i, s_j, a)$ represent the probability that the searcher transitions to state s_j by performing action a in state s_i. Similarly, the entries of the reward matrix $\mathbb{R}(s_i, s_j, a)$ represent the transition reward from state s_i to state s_j after performing action a.

When some components of the state are not fully observable, we have a Partially Observable Markov Decision Process (POMDP). Similar to MDPs, a POMDP is represented as a tuple $(\mathbb{S}, \mathbb{A}, \mathbb{T}, \mathbb{R}, \mathbb{Z}, \mathbb{O})$ where $\mathbb{O}$ is the set of observations, and $\mathbb{Z}$ is a matrix with entries $\mathbb{Z}(o, s, a)$ that gives the conditional probability of observing o after performing action a and moving to the state s.

The MDP and POMDP are used to formulate optimization problems where the goal is to find an optimal policy, i.e., a sequence of actions, such that the cumulative reward of the agent is maximized. In the case of probabilistic search problems, the goal is to find a search strategy (policy) such that the capture probability is maximized. In Chapter 6, we present the formulation of our random walker search problem as a POMDP: We specify the state space as well as the reward function. We refer the interested reader to [66, 67] for some well-known methods to solve for the optimal policy (such as value iteration, policy iteration and reinforcement learning).

It is known that POMDPs suffer from curse of dimensionality, i.e., very large state space [68]: When the underlying MDP has N states, the belief is in a N-dimensional space. One approach to overcome the issue is to use a sampling-based algorithm to find the optimal policy instead of computing the best policy in the whole belief space [69,70]. A well-known point-based algorithm is SARSOP which stands for Successive Approximations of the Reachable Space under Optimal Policies [71, 72]. In this approach a set of points are sampled from the belief space and these samples are used as an approximate representation of belief. Exploiting these samples, a belief tree is maintained with the initial belief as the root node. Moreover, the subtrees that will never be visited by the optimal policy are pruned out. In order to estimate the optimal value function V^* a piece-wise linear lower bound $\underline{V}$ and also an upper bound $\overline{V}$ are maintained. To improve these bounds, the algorithm applies Bellman backup operation until convergence is achieved. For our purpose in Chapter 6, we use an implementation of the SARSOP algorithm called Approximate POMDP Planning (APPL) toolkit which is available in [73].

3.2.2 Mixed Observability Markov Decision Process (MOMDP)

An alternative approach to tackle the large state space of our problem is to formulate the problem as a Mixed Observability MDP [58]. In this formulation, the state components that are fully observable are separated from the ones that are partially observable. As a result, the belief is maintained on a smaller set of variables, and the size of the state space can be reduced significantly. More specifically, a MOMDP is specified as a tuple $(\mathbb{X}, \mathbb{Y}, \mathbb{A}, \mathbb{O}, \mathbb{T}_x, \mathbb{T}_y, \mathbb{R}, \mathbb{Z})$ where $\mathbb{X}$ represents the set of fully observable components, $\mathbb{Y}$ represents the set of partially observable components, and $\mathbb{A}, \mathbb{O}$ are the set of actions and observations respectively. The function $\mathbb{T}_x(x, y, a, x')$ represents the probability that after taking action a in state (x, y) the fully observable state component x makes a transition to the new value x'. Similarly, the function $\mathbb{T}_y(x, y, a, y')$ gives the probability that the partially observable component has the new value y'. The belief is then represented as (x, b_y) where b_y is the belief defined only for the y component.

As an example, consider a probabilistic search problem on a graph. Suppose that the searcher detects the target when both of them are on the same node of the graph. In addition the players cannot observe each other's location unless they are on the same node. In this problem, the location of the searcher is fully observable while the target's location is the unobservable component. This search problem is formulated as a

MOMDP in Chapter 6 taking advantage of the separation between the fully observable and partially observable components of the state.

In this chapter, we covered the basic technical tools and the necessary background to solve the lion and man game (pursuit-evasion games) and the random walker search problem (probabilistic search problem). In the next chapter, we will discuss the lion and man game with line-of-visibility. Then, we focus on the full-visibility variant of the lion and man game in Chapter 5. The random walker search problem is presented in Chapter 6.

Chapter 4

Pursuit-Evasion Games with Limited-Visibility

Let us now start our detailed study of the search problem[1]. In the following two chapters, we focus our attention on the adversarial search problem, i.e., pursuit-evasion games. In this chapter, we study a version of the pursuit-evasion game with partial knowledge of the evader's location; in particular, when the pursuer has only line-of-sight vision. That is, the pursuer can see the evader only if the line segment connecting them is free of obstacles. This variant models robotics applications where the pursuer is a robot equipped with a camera or a laser scanner. In the next chapter, we investigate different pursuit-evasion games when the pursuer has complete knowledge of the evader's location.

Our goal in this chapter is to design a pursuit strategy such that the pursuer can capture the evader. Consequently, with the limited vision power, the pursuer has to first find the evader when it disappears and then move toward the evader to capture it. We show that despite this limitation, the pursuer can capture the evader in any **monotone polygon** in finite time.

A simple polygon is called monotone with respect to a line l if for any line l' perpendicular to l the intersection of the polygon with l' is connected [74]. In this work, without loss of generality we consider x-monotone polygons. For the pursuit-evasion

[1] The material in this chapter appears in [12].

game with visibility in monotone polygons, we present a pursuit strategy which successfully combines search and capture and guarantees that the evader will be captured after a finite number of steps.

In monotone polygons, merely searching for the evader is straightforward: the evader can be found for example by moving along the shortest path that connects the leftmost vertex to the rightmost vertex. This is because every point inside the polygon is visible from a location on this path and the evader cannot move into a "cleared" region without revealing itself.

Similarly, the capture strategy is simple when the pursuer knows the location of the evader at all times: it can capture the evader by starting from the leftmost vertex O_L and performing the lion's strategy (Chapter 3): that is to move toward the evader along the shortest path that connects O_L to the evader's current location. The distance of the pursuer from O_L, defined as the length of the shortest path from O_L to the pursuer's location, provides a natural notion of "progress" which is monotonically increasing in this full visibility setting[2].

What is not obvious is whether such progress can be maintained when the pursuer has to search for the evader when it disappears. If the pursuer ends up retreating after a search, the evader might have a strategy in which the pursuer oscillates between search and progress and the game can last forever. We show that this cannot happen. In particular, we show that the pursuer can successfully combine search and making progress toward capture in monotone polygons. Further, we show that search without risking progress can be achieved with a deterministic strategy. We are not aware of any other results which combine these two objectives for a single pursuer while providing guarantees about the outcome of the game. The randomized strategy proposed for the general simply connected polygons [75], where the pursuer guesses the hiding vertex of the evader, provides exponential capture time[3]. In this work, however, we present a deterministic pursuit strategy which reduces the capture time to a quantity that is polynomial in the number of vertices and the diameter of the polygon. An interesting feature of our strategy is that the pursuer's distance from O_L is not increasing monotonically. Nevertheless, we show that the pursuer can push the evader further to the

[2] This argument works in any simply connected polygon [75].
[3] It is an open problem whether this bound is tight or not.

right after a finite number of steps. To do this, we introduce a new measure of progress for the pursuer which is more sophisticated than its distance from O_L.

Our results provide a step toward understanding the pursuit-evasion game with visibility constraints in general polygons. An important question is to find the class of polygons in which a single pursuer suffices to capture the evader[4]. We show that monotone polygons are included in this class.

This chapter is organized as follows. In Section 4.1, we present a precise description of the game model we use in this chapter as well as the notation throughout this chapter. Section 4.2 provides an overview of the pursuit strategy. In Section 4.3, we present the tools used to show that the strategy guarantees capture. An illustrative example is presented in Section 4.4 where we give an example scenario to show the strategy in action. The details of the pursuit strategy is presented in Section 4.5 and Section 4.6. In particular, Section 4.5 covers the search component of the strategy to find the evader when it is invisible, and Section 4.6 presents the strategy for actually making progress. For each part of the strategy, we also present the complete algorithm in the form of pseudocode. In particular, we provide the input configuration which describes the conditions that must be satisfied before the pursuer starts the corresponding sub-strategy as well as the exit configuration that the pursuer guarantees as it switches to the next sub-strategy. We present the analysis of the capture time in Section 4.7. We present the detailed proof of the technical lemmas as well as the properties of monotone polygons in Section 4.8. The concluding remarks are discussed in Section 4.9.

4.1 Game Model and Notation

We now formally define the game. We refer to the pursuer's and the evader's location at time-step t as $p(t)$ and $e(t)$ respectively. When the time is obvious from the context, we use p and e. Our *Game Model* is as follows: (1) The players move alternately in turns. (2) Each turn takes a unit time step. (3) In each turn the players can move along a line segment of length at most one to a point visible to themselves. (4) The evader has global visibility i.e. it knows the location of p at all time steps. However, the pursuer sees the evader only if the line segment joining the two is not blocked by

[4] It should also be noted that capture using only deterministic strategies remains an open question.

the boundary of the polygon. Note that since p has a deterministic strategy, the evader can simulate the pursuer's moves and hence it knows the location of p at all time steps. (5) The pursuer captures e if at any time, the distance between them is less than or equal to one (the step-size) while p can see e, see Fig. 4.1(c).

Without loss of generality, we assume that the game takes place in an x-monotone polygon Q. Recall that for an x-monotone polygon, the intersection of all vertical lines with the polygon is a connected segment. The leftmost vertex and the rightmost vertex are denoted by O_L and O_R respectively. The boundary of the polygon connects these vertices by two x-monotone chains denoted by $Chain_L$ and $Chain_U$, see Fig. 4.1(a).

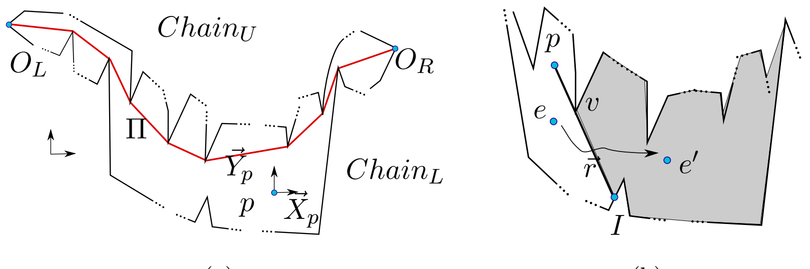


Figure 4.1: (a) An x-monotone polygon. (b) The pocket $pocket(v, \vec{r})$ is the shaded sub-polygon. (c) Since e is not visible, capture condition is not satisfied.

We refer to the segment between points u and v as uv. Whenever direction is also important we refer to the ray pointing from u to v as $\vec{uv}$. We define a local reference frame whose origin coincides with p. Its axes $\vec{X}_p$ and $\vec{Y}_p$ are parallel to the axes of the reference frame, see Fig. 4.1(a). We refer to the boundary of Q as ∂Q, and the number of vertices in Q as n. The shortest path between the two points u and v is denoted by $\pi(u, v)$, and the length of $\pi(u, v)$ is denoted by $d(u, v)$. The diameter of the polygon is $D = max\ _{u,v \in Q}\ d(u, v)$. The shortest path tree rooted at the point o in Q is defined as $\cup_{v \in V} \pi(o, v)$ where V denotes vertices of Q. For a point p inside the polygon, the parent of p, denoted by $parent(p)$, is the first vertex on the shortest path $\pi(o, p)$ from p to o. In the rest of the paper, we consider the shortest path tree rooted at $o = O_L$. We denote $\pi(O_L, O_R)$ by Π (Fig. 4.1(a)). For simplicity we denote $d(O_L, p)$ by $R(p)$ for a point $p \in Q$.

Throughout the chapter, we refer to the x coordinate of a point p as $x(p)$. Similarly, the y coordinate is denoted by $y(p)$.

Suppose that it is the evader's turn to move. See Fig. 4.1(b). Imagine that the pursuer and the evader can see each other before the evader's move but the evader disappears behind a vertex v after moving to e'. Let $\vec{r}$ be a ray originating from p and passing through v. Let I be the intersection of this ray with the polygon. The sub-polygon which contains e and is bounded by the ray $\vec{r}$ plus the boundary of the polygon from v to I is called a *pocket* [75]. The ray $\vec{r}$ is called the entrance of the pocket and the pocket is referred to as $pocket(v, \vec{r})$.

Remark 1. *In the rest of the chapter, we refer to the pocket that the evader is hidden inside of as $pocket(v, \vec{r})$ (also the* contaminated region*) where $\vec{r} = \vec{pv}$ and p is the location of the pursuer at the time that the evader has disappeared behind the vertex v. See Fig. 4.1(b).*

We are now ready to present our capture strategy. We start with an overview of the strategy discussing the main ideas.

4.2 Monotone Polygon Capture Strategy: Overview

In this section, we start with a high level description of the pursuer's strategy. The details of the strategy is provided in the following sections.

We start with an example which demonstrates the difficulty of designing capture strategy for a pursuer that has limited vision power. Fig. 4.2 provides some intuition. In this example, we also show the importance of pursuer's notion of progress which is required to prove that the proposed pursuer strategy will guarantee capture in finite time. Suppose that the pursuer's notion of progress is its x coordinate and moreover let $x(p) \leq x(e)$ be the invariant that the pursuer tries to maintain. Therefore, if the pursuer's strategy guarantees that the x coordinate of p is increased after finite time, the evader will be captured in finite time. Now, suppose that p is following e by lion's move with respect to O_L and e disappears behind v in the shaded pocket. Hence p has to search for e. A careless search strategy may result in losing the progress that p has made so far as follows. If the pursuer first visits v_2, then the evader hidden at v_1 will escape to v and re-enter the previously "cleared" region i.e. the set of points p such that $x(p) \leq x(p)$ defined by pursuer's notion of progress. Likewise, if the pursuer first visits v_1, the evader hidden at v_2 can re-contaminate the cleared region. Consequently, the evader can hide in the same portion of the polygon infinitely many times and hence avoid capture (against this naive pursuit strategy).

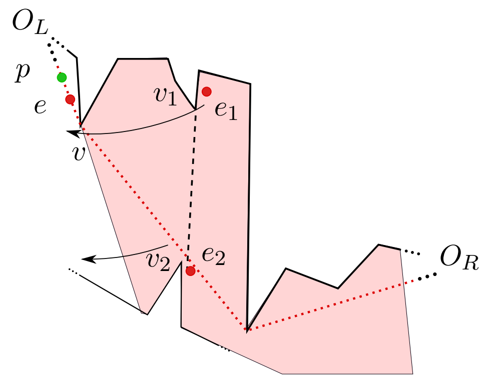


Figure 4.2: A difficult situation for p: e can re-contaminate the cleared region depending on how p enters the pocket.

We will present a pursuit strategy called *Monotone Polygon Capture (MPC)* strategy which guarantees capture. In this strategy, we partition the monotone polygon into sub-polygons called the *critical sub-polygons*. The pursuer clears these sub-polygons from the left to the right. That is, p ensures that the evader cannot re-contaminate the cleared portion and hence it will be captured.

The state diagram of the MPC strategy is given in Fig. 4.3, and a high level description is presented in Algorithm 1. The strategy consists of three states: *Search, Guard* and *Extended Lion's Move* denoted by S, G, and L respectively. Initially, p is at O_L. It starts by the S state if e is invisible, and the L state otherwise. Whenever e disappears, p switches to the search state to find it.

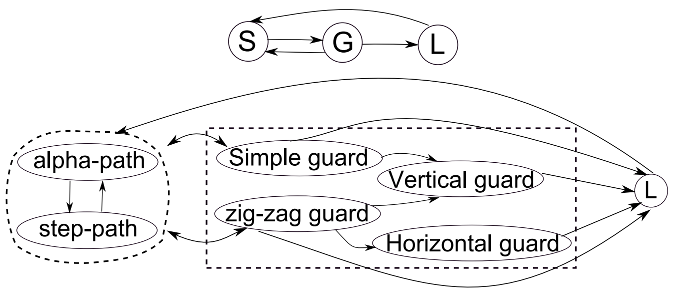


Figure 4.3: The state diagram for the MPC strategy. The sub-states in S and G are shown at the bottom.

Algorithm 1: Monotone Polygon Capture Strategy

```
1 repeat
2     if $e$ is invisible then
3         do Search strategy until $e$ is found;
4     else if $e$ is visible and $p$ is not on $\pi(O_L, e)$ then
5         do Guard strategy until $p$ is on $\pi(O_L, e)$, or $e$ disappears;
6     else if $e$ is visible and $p$ is on $\pi(O_L, e)$ then
7         do Extended Lion's Move strategy until $e$ is captured, or $e$ disappears;
      end
  until $e$ is captured;
```

When e is found as a result of the S strategy, the pursuer performs the *guard* strategy in order to establish the extended lion's move with respect to O_L. Recall that p has to be on $\pi(O_L, e)$ in order to be able to perform the extended lion's move. Therefore in the guard state p tries to catch up with $\pi(O_L, e)$ since p might not be on $\pi(O_L, e)$ at the time that e is found. After p catches up with $\pi(O_L, e)$, the pursuer follows it by extended lion's move. During the lion's move state or the guard state, e might disappear. At this time p switches to the search strategy. The sequence of state transitions is $((SG)^*L)^*$ and the loops $(SG)^*$ and $L(SG)^*L$ are possible. In the following, we refer to the S state and its following G state as the *combined Search/Guard* state i.e. (SG).

To prove that our proposed strategy guarantees capture, it is necessary to show that these loops terminate after finite time. To do so, we define a *reference vertex*, denoted by p_{ref}, which is a vertex of the polygon. We show that p maintains the invariant that e is to the "right" of p_{ref} at the beginning of a combined search/guard state (SG) or an L state. The pursuer makes progress by updating p_{ref} to the "right" after finite number of time steps. Consequently, the evader is confined in a smaller region and hence it will be captured.

We define "right" of a vertex v as the half-plane to the right, below or above v based on the structure of the monotone polygon. We denote the half-plane associated with the vertex v by $h(v)$. Figure 4.5 illustrates examples of these half-planes. Then, the invariant is that the evader is forced to remain inside $h(p_{\text{ref}})$ at the beginning of a combined search/guard state (SG) or an L state (otherwise it will be captured), and the progress is to update p_{ref} to a new point p'_{ref} such that $p'_{\text{ref}} \in h(p_{\text{ref}})$.

It is worth emphasizing that the aforementioned invariant is guaranteed only at the beginning of a (SG) state or an L state. During the guard state, the evader can exit $h(p_{\text{ref}})$ and re-contaminate the region before p_{ref}. At this time, the pursuer switches to the *Vertical Guard* or the *Horizontal Guard* sub-state in order to push e back to $h(p_{\text{ref}})$ and recover its progress. See the state diagram in Fig. 4.3.

For ease of reference, we now list the terminology that is crucial in this chapter. We will present the formal definition of these variables in the following sections.

- The reference vertex p_{ref} which is used to track the progress. The evader is guaranteed to be to the right of p_{ref}.
- Half-planes associated to a vertex v denoted by $h(v)$ which denotes right of a vertex v used to track progress.
- The auxiliary vertex denoted by p_{aux}, which we introduce in Section 4.6.2, is a local variable used to track the progress.

Let us next present the details of the invariants and the pursuer's notion of progress.

4.3 Definitions, Invariants and the Notion of Progress

The critical sub-polygons that partition the monotone polygon are defined as follows.

Definition 4.3.1 (Critical Sub-polygons)**.** *Let Π be the shortest path from O_L to O_R and denote the vertices on Π by $\{v_0, ..., v_i, ...\}$. Let s_{i-1} be the slope of the edge $v_{i-1}v_i$ and $\delta s_i = s_i - s_{i-1}$. Then, the* critical vertices *are those vertices on Π on which either s or δs changes sign. For example, in Fig. 4.4, the vertices v_i, v_k and v_j are the critical vertices. For a critical vertex $p = v_i$, we assign $\vec{Y}_p$ if in $p = v_i$, the values s_i and δs_i have different signs and $\vec{X}_p$ if they have the same signs. Then, each two consecutive critical vertices, say v_i and v_k, define a* critical sub-polygon *given by the sub-polygon formed by ∂Q and the rays assigned to v_i and v_k. See Fig. 4.4. Depending on the sign of s and δs in the critical sub-polygons, we get four types of critical sub-polygons: Type (1) when $s < 0$ and $0 < \delta s$, Type (2) when $0 < s$ and $0 < \delta s$, Type (3) when $0 < s$ and $\delta s < 0$, Type (4) when $s < 0$ and $\delta s < 0$.*

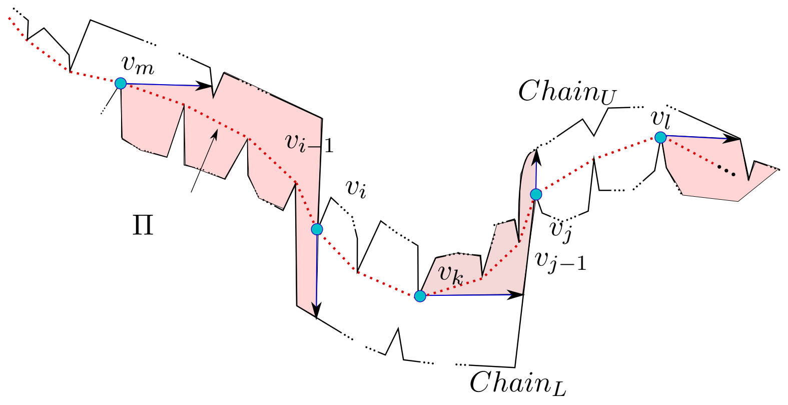


Figure 4.4: The path Π is shown in dots. The critical sub-polygon defined by v_m and v_i is type 4, v_i and v_k is type 1, v_k and v_j is type 2, and v_j and v_l is type 3.

The critical sub-polygons allow us to enumerate all possible configurations between p, e and the structure of P since our proposed strategy is symmetric in different types of these critical sub-polygons.

Remark 2. *Throughout the chapter, we present the strategy for the case that v, the vertex that defines the hiding pocket $pocket(v, \vec{r})$, is inside the 1^{st} type critical sub-polygons. The strategies for the other types are symmetric which we specify them in the form of Remarks.*

Without loss of generality we assume that at the beginning of the search state the pursuer is at v. This is possible because, after e disappears, the pursuer moves toward v along the straight line pv until it reaches v. If in the meantime e appears, the pursuer resumes the last state's strategy, i.e. the G state or the L state. Note that all preconditions of this state are still satisfied.

We now present an important property of Q, and then explain how the pursuer makes progress. Let us first define the *half-plane* of a vertex.

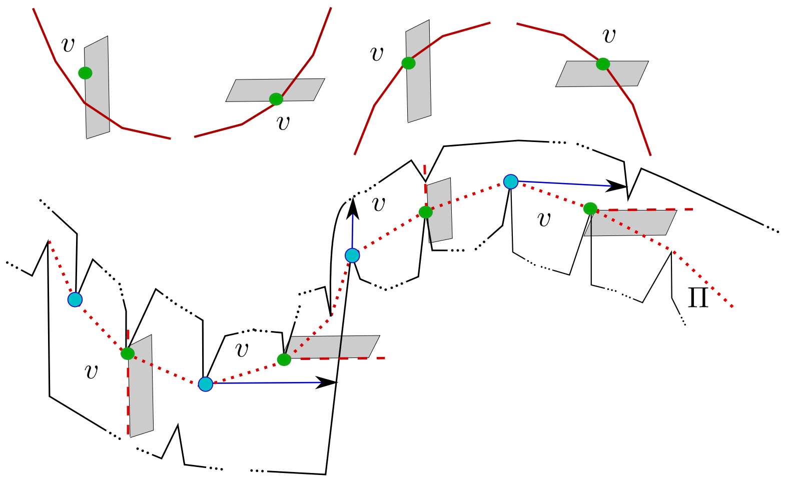


Figure 4.5: Bottom) The dot path is Π. The half-planes associated to a vertex v. Top) From left to right are types 1 to 4. The path Π and $h(v)$ are shown.

Definition 4.3.2 (The half-plane of a vertex)**.** *For a vertex $v \in Q$, the* open half-plane, *denoted by $h(v)$, is defined as the set of points to the right of v if v is inside the 1^{st} or the 3^{rd} type critical sub-polygons. For the 2^{nd} type, $h(v)$ is the half-plane above v i.e. the points p with $y(v) < y(p)$. Finally for the 4^{th} type, $h(v)$ is the half-plane below v i.e. the points p with $y(v) > y(p)$. The corresponding closed half-planes are denoted by $h[v]$. See Fig. 4.5.*

The following property of monotone polygons is crucial in this paper which we prove in Section 4.8.1.

Property 4.3.3. *Consider the critical sub-polygons defined in Definition 4.3.1. Let v be any vertex in Q and refer to Fig. 4.5.*

- *Suppose that v is inside a critical sub-polygon of the* 1^{st} *type,* $v \in Chain_U$*, and the slope of the edge between* $parent(v)$ *and* v *is negative. Then for all points* $p \in h(v)$*, we have* $R(v) < R(p)$*.*

- *Suppose that v is inside a critical sub-polygon of the* 3^{rd} *type,* $v \in Chain_L$*, and the slope of the edge between* $parent(v)$ *and* v *is positive. Then for all points* $p \in h(v)$*, we have* $R(v) < R(p)$*.*

- *Suppose that v is inside a critical sub-polygon of the* 2^{nd} *type,* $v \in Chain_U$*, and the slope of the edge between* $parent(v)$ *and* v *is positive. Then for all points* $p \in h(v)$*, we have* $R(v) < R(p)$*.*

- *Suppose that v is inside a critical sub-polygon of the* 4^{th} *type,* $v \in Chain_L$*, and the slope of the edge between* $parent(v)$ *and* v *is negative. Then for all points* $p \in h(v)$*, we have* $R(v) < R(p)$*.*

The reference vertex p_{ref}: The invariant that the pursuer maintains and its notion of progress are defined based on the vertex p_{ref}. Let v be the vertex that defines the hiding pocket *pocket*$(v, \vec{r})$. Initially $p_{\text{ref}} = O_L$. The vertex p_{ref} is defined such that $v \in h[p_{\text{ref}}]$ and $p_{\text{ref}} \in Chain_U$. The pursuer updates p_{ref} at the beginning of an S state which is after a G state (Section 4.6). Specifically, when v belongs to $Chain_U$ we have $p_{\text{ref}} = v$. In case that $v \in Chain_L$ the vertex p_{ref} is set to another vertex from the upper chain as explained in Section 4.6.1 and Section 4.6.2.

Remark 3. *When p is sweeping the* 3^{rd} *or the* 4^{th} *type critical sub-polygons* $p_{\text{ref}} \in Chain_L$*. Also in the* 2^{nd} *type critical sub-polygon* $p_{\text{ref}} \in Chain_U$*.*

Next we present the invariants that p maintains during the game. At the beginning of a combined search/guard state (SG) or an L state we have:

- **Invariant (I1)** $e \in h(p_{\text{ref}})$ and $p \in h[p_{\text{ref}}]$. Consequently $R(p_{\text{ref}}) \leq R(p)$ and $R(p_{\text{ref}}) < R(e)$ (Property 4.3.3).

- **Invariant (I2)** whenever e is invisible, it is inside *pocket*$(v, \vec{r})$ where v is the leftmost vertex of *pocket*$(v, \vec{r})$ and $v \in h[p_{\text{ref}}]$.

The pursuer achieves one of the following notions of progress after a finite number of time-steps. Consider two consecutive combined (SG) states, and let t and t' be the time-steps that p starts them correspondingly. Suppose that p_{ref} and p'_{ref} are the old and the new reference vertices at t and t' respectively. Also let v and v' be the old and new vertices which define the pockets $pocket(v, \vec{r})$ and $pocket(v', \vec{r'})$ respectively. Then:

- **Progress (P1)** either the pursuer updates p_{ref} to a new vertex p'_{ref} so that $p'_{\text{ref}} \in h(p_{\text{ref}})$ and consequently $R(p_{\text{ref}}) < R(p'_{\text{ref}})$ (Property 4.3.3),
- **Progress (P2)** or p_{ref} remains the same and the pursuer updates the contaminated region $pocket(v, \vec{r})$ to $pocket(v', \vec{r'})$ such that $v' \in h(v)$.

Our main result is the following theorem which we prove in Section 4.7:

Theorem 4.3.4. (Progress) *Suppose that Q is a monotone polygon. Then the pursuer by following the MPC strategy can capture the evader in $O(D^{13}n^7)$ steps where n is the number of vertices of Q and D is the diameter of Q.*

4.4 An Illustrative Example

Let us present an example which illustrates the MPC strategy (Fig. 4.6). Initially p is at O_L, e is visible and $p_{\text{ref}} = O_L$ (Fig. 4.6(a)). Therefore, p can follow e by extended lion's move (Fig. 4.6(b)). This continues until e disappears behind the vertex v. The pursuer moves toward v but in the meantime the evader also moves to an unknown location inside $pocket(v, \vec{r})$ say e_5 (Fig. 4.6(c)). The pursuer updates p_{ref} to v since v, the pocket vertex, is on the upper chain. It also switches to the search state and moves along the dotted line in order to find e (Fig. 4.6(c)). By moving along this path the pursuer finally finds e at p_6 (Fig. 4.6(d)). Next, the pursuer switches to the guard state in order to catch up with $\pi(O_L, e)$ and re-establish the extended lion's move state. To do so, it defines a vertex called p_{aux} which is inside $h[p_{\text{ref}}]$[5] (Fig. 4.6(d)). Then, p moves toward p_{aux} until e crosses the ray shot from p in direction of p_{aux} to p e.g. in p_7 and e_7 (Fig. 4.6(e)). At this time, the pursuer follows e by lion's move with respect to p_{aux} (Fig. 4.6(e)). During this lion's move the players cross the vertical line passing through

[5] Note that p_{aux} can be the same as p_{ref}, but in this example they are different.

p_{aux} to the left and enter the region $(Q - h[p_{\text{aux}}])$ (Fig. 4.6(e)). In other words, e recontaminates the region to the left of p_{aux}[6]. At this time, the pursuer switches to the vertical guard sub-state in order to push the evader back to $h(p_{\text{aux}})$. The strategy in this state ensures that the evader cannot enter the shaded region in Fig. 4.6(e). The pursuer does this by performing the lion's move with respect to c which is the center of the circle that passes through p_9 and I. The result is that p catches up with $\pi(O_L, e)$ inside $h(p_{\text{aux}}) \subseteq h(p_{\text{ref}})$ after finite time (Fig. 4.6(f) at p_{10}). Hence p can again follow e by the extended lion's move while it has pushed p_{ref} and e to the right. Similarly, the pursuer keeps making progress by updating p_{ref} to the right and ensuring that $e \in h(p_{\text{ref}})$.

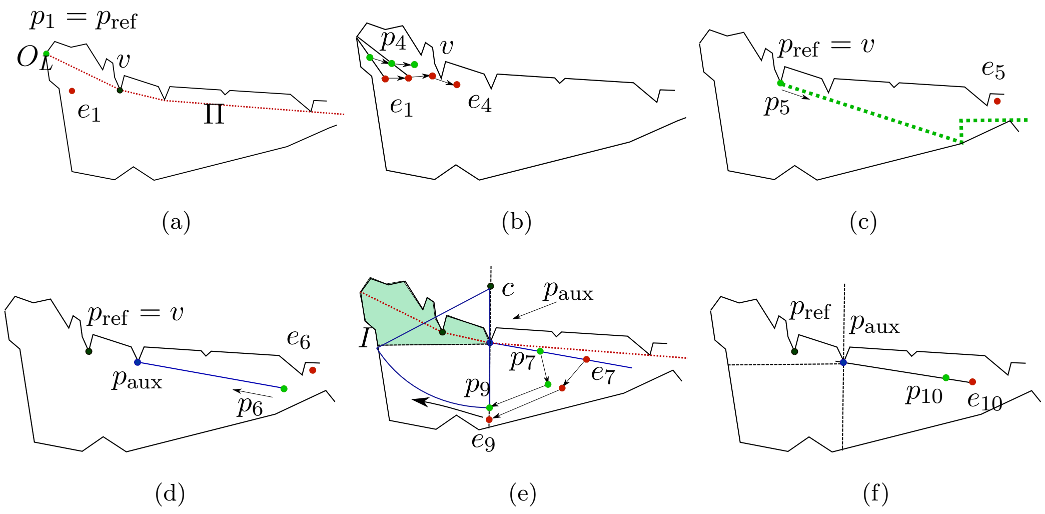


Figure 4.6: An instance of the game. The path Π is shown in dots. (a) Beginning of the game. (b) p follows e by extended lion's move. In the fourth time step the evader hides behind v. (c) By the time that p arrives at v the evader has moved to e_5. The pursuer updates p_{ref} to v, and searches for e by moving along the dotted path. (d) The pursuer finds e at p_6. It defines p_{aux} and moves toward it. (e) As e crosses the line from p_{aux} to p, the pursuer follows it by lion's move with respect to the center p_{aux} (p_7 and e_7). When e moves to the left of p_{aux} at e_9, p follows it by lion's move with respect to c. (f) Finally, p reaches $\pi(O_L, e)$ and switches to extended lion's move inside $h(p_{\text{ref}})$.

We now present the details of each state starting with the *search* component.

[6] When $p_{\text{aux}} = p_{\text{ref}}$, this is equivalent to violating the invariant (I1).

4.5 Search State

When e disappears, the pursuer performs the search strategy in order to find e. Suppose that e is hidden inside $pocket(v, \vec{r})$. At the time that e disappears behind v, the pursuer walks toward the blocking vertex v. Hence without loss of generality we can assume that at the beginning of the S state p is at v, the pocket vertex.

In order to find e, the pursuer moves along a path. The following observation ensures that as long as this path is monotone in the x-axis direction, p finds e after finite time.

Observation 4.5.1. *Let p_1 and p_2 be two points inside the polygon where $x(p_1) < x(p_2)$ and suppose that $x(p_1) < x(e)$. Suppose that the pursuer moves from p_1 to p_2 along any (continuous) arbitrary path. Then, if the pursuer reaches p_2 and the evader is still invisible, it must be that $x(p_2) < x(e)$. Otherwise if e becomes visible before the pursuer reaches p_2, then at the time that e is found it must be that $x(p) < x(e)$.*

Let us refer to the path that the pursuer moves along in order to find the evader as the *search path*. The search path consists of two types of paths, the *α-path* and the *step-path*. Intuitively, the α-path periodically touches the upper chain while the step-path touches the lower chain[7]. The pursuer uses these touching points as landmarks that it has to prevent the evader from re-contaminating the region to the left of those landmarks. In fact, the reference vertex p_{ref} is set to these landmarks in some cases. As we will see shortly, the search path is composed of horizontal lines, vertical lines and lines with negative slope. An example is depicted in Fig. 4.6(c). The pursuer exploits this slope in order to guarantee progress in the situations that the evader forces the pursuer to retreat to the left of the landmarks. For example, in Fig. 4.6(e), if this slope was zero, i.e. p_7 was at the same y-coordinate as the landmark p_{aux}, and the evader has disappeared below the pursuer e.g. at e_7, then the pursuer had to retreat along the horizontal line all the way back to I without making any progress. However, the slope provides a lower bound on the distance between the pursuer and the landmark p_{aux} at p_9 which translates into guaranteed progress.

In the following, we will first present the definition of the α-path and the step-path, and afterwards we explain how the search path is built from them.

[7] This is when p is inside the first type critical sub-polygons. We explain the modifications required for other types in Remark 4.

We define the α-*segments* as follows. The segments that make angle $-\alpha$ with the x-axis are called the α-*segments*. As we will see in Section 4.6.3, the angle α is used to bound the time spent in the G state.

Definition 4.5.2. *The angle* α *is chosen as the minimum of the two angles* $\psi_1 = \arcsin \frac{1}{D}$ *and* $\psi_2 = (\frac{\pi}{2} - 2\arctan\frac{1}{2})$ *where* D *is the diameter of the polygon.*

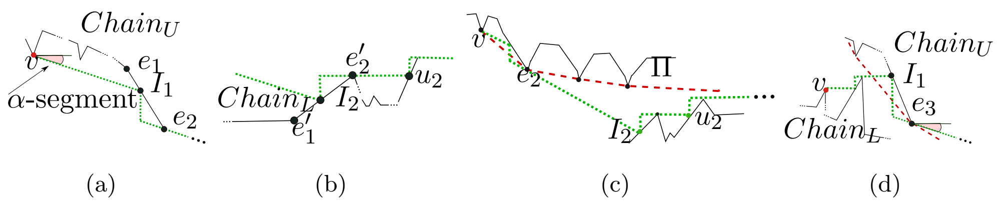


Figure 4.7: The search path is shown in dots. The path Π is shown in dashed line. (a) A single α-step. The portion of the search path from v to e_2 is one α-step. (b) A single step. The portion of the search path from I_2 to u_2 is one step. Note that I_2 is the floor point. (c) Two α-steps and two steps are shown. (d) When v, the pocket vertex, is from the lower chain, the search path starts by the step-path.

The α-steps and the α-path: The α-path is composed of a number of α-*steps*. A single α-step is composed of an α-segment followed by a vertical segment and then another α-segment. For example, in Fig. 4.7(a) the portion of the search path from v to e_2 is an α-step. More specifically, let $e = e_1e_2$ be the edge on ∂P that the first α-segment intersects and let I_1 be the point of intersection. The edge e can be either on $Chain_U$ or $Chain_L$.

If $e \in Chain_U$, then the α-step continues along the vertical segment passing through I_1 until this segment intersects the α-segment passing through e_2. The α-step then continues along this α-segment until it reaches e_2. We refer to this part of the search path from v to e_2 as a single α-step (Fig. 4.7(a)).

If $e \in Chain_L$ then the α-step will be followed by the step-path described below. See Fig. 4.7(b). In summary, the search path continues along a number of α-steps, which together are called the α-path, until it hits the lower chain in which case it continues along the step-path. See Fig. 4.7(c).

The steps and the step-path: The step-path can be divided into a number of *steps*. A single step is composed of a vertical segment followed by a horizontal segment. For example, in Fig. 4.7(b), the portion of the search path from I_2 to u_2 is one step. Detailed definition of an step is the following. Let e_2I_2 be the α-segment from the α-path that intersects the lower chain (Fig. 4.7(c)). Also, let $e' = e'_1e'_2$ be the corresponding edge on $Chain_L$ (Fig. 4.7(b)). Then e_2 is called the *ceiling* point and I_2 is called the *floor* point. The step-path starts at the floor point I_2, continues along the vertical line passing through I_2 until this vertical line intersects the horizontal line passing through e'_2 and then continues along this horizontal line until it hits ∂P at u_2. This portion of the search path from I_2 to u_2 is referred to as a single *step* (Fig. 4.7(c)).

The search path: Finally, the search path is composed of the α-path and the step-path as follows. If $v \in Chain_U$, the search path starts by the α-path, otherwise if $v \in Chain_L$, it starts by the step-path. See Fig. 4.7 parts (c) and (d) respectively. Suppose that $v \in Chain_U$. Then, the search path continues along the α-path until it hits the lower chain. As it hits the lower chain it continues along the step-path until it hits the upper chain in which case it continues again along the α-path, and so on. This switch between the step-path and the α-path is depicted in the state diagram of Fig. 4.3. As an example, note that in Fig. 4.7(d) from v to I_1 we have the step-path and after e_3 we have the α-path.

Remark 4. *The general rule for the search path for other types of critical sub-polygons is as follows. The* 1^{st} *and the* 3^{rd} *types include both the step-path and the α-path with angles $-\alpha$ and $+\alpha$ respectively. The* 2^{nd} *and the* 4^{th} *types only have the step-path. The direction traveled parallel to the y-axis during the step-path and the α-path, is the same as the sign of δs and the sign of s respectively (i.e. positive sign means upward, negative sign means downward).*

Lemma 4.5.3. *The pursuer finds e after at most $O(nD)$ steps where n is the number of vertices of the polygon and D is the diameter of the polygon. Moreover, at the time that the evader is found we have $x(v) \leq x(p) < x(e)$ where v is the pocket vertex. Therefore, at the end of the search state we have $p \in h[p_{\text{ref}}]$ and $e \in h(p_{\text{ref}})$.*

Proof. We first bound the length of the search path that will be used to search the whole polygon. Consider the smallest bounding box that encompasses the polygon. Let H and

W denote the height and the width of this bounding box respectively. According to the triangle inequality theorem, the length of the search path is less than the displacement of p along the x-axis and the y-axis as the pursuer moves along the search path. First, the total displacement along the x-axis is less than W since the x coordinate of the points on the search path never decreases. Second, associated with each vertex, the search path traverses in the direction of the y-axis at most once upward and at most once downward (Fig. 4.7(d)). Therefore, the total displacement along the y-axis is $2nH$. Thus, the length of the search path is less than $W + 2nH = O(nD)$.

According to Observation 4.5.1, at the time that e is found $x(v) \leq x(p) < x(e)$ since the x coordinate of p is increasing as it moves along the search path and at the beginning of the search state p is at v. Consequently, we have $p \in h[p_{\text{ref}}]$ and $e \in h(p_{\text{ref}})$ since $v \in h[p_{\text{ref}}]$ (invariant (I2)). $\square$

4.6 Guard State

After p finds e, it starts the *Guard* state. The purpose of the guard strategy is to establish the extended lion's move. The extended lion's move is possible only when p is on $\pi(O_L, e)$. Therefore, in the guard state the pursuer has to reach $\pi(O_L, e)$. In the meantime, th evader is also moving and thus the pursuer has to preserve the progress it made so far.

At the beginning of the guard state we have $x(p) < x(e)$ (Lemma 4.5.3,). At this time, based on the quadrant that e has appeared in, the pursuer starts different sub-states, either the *zig-zag guard* strategy or the *simple guard* strategy as shown in Algorithm 2 and the state diagram of Fig. 4.3.

1. If at the beginning of the G state, e is inside the fourth quadrant of p (lines 1-5 in Algorithm 2): the pursuer performs the zig-zag guard strategy.

2. Otherwise, if e is inside the first quadrant of p (lines 6-11 in Algorithm 2): the pursuer starts by simple guard sub-state.

Let p_{ref} be the current reference vertex used for tracking progress. The pursuer's ultimate goal is to maintain the invariant and the notion of progress. That is to ensure that e is to the right of p_{ref} i.e. inside $h(p_{\text{ref}})$. However, during zig-zag guard sub-state and simple guard sub-state, the evader can re-contaminate the region to the left of p_{ref} i.e. the region $(Q - h[p_{\text{ref}}])$. The evader does this by crossing the vertical line passing through p_{ref} to the left, and entering $(Q - h[p_{\text{ref}}])$. See line 5 and line 11 in Algorithm 2. This violates the invariant (I1). At this time, the pursuer switches to the *vertical guard* sub-state in order to push the evader back to the right of p_{ref} (line 24 in Algorithm 2). Hence at the end of the vertical guard state the invariant (I1) will be re-established.

At the end of the guard state, as the pursuer switches to the next state, it guarantees progress (P1) or (P2). Specifically, if progress (P1) is achieved the pursuer updates the reference vertex p_{ref}. See lines 15- 21. We show that the new reference vertex p'_{ref} is inside $h(p_{\text{ref}})$ and so the evader is pushed further to the right since the invariant $e \in h(p'_{\text{ref}})$ is also valid (Lemma 4.6.1 and Lemma 4.6.2). We proceed by presenting the details of each sub-state, the zig-zag guard, the simple guard and the vertical guard.

Remark 5. *If at the beginning of the G state, the players are inside a critical sub-polygon of the* 2nd *type, the* 3rd *type or the* 4th *type, then:*

1. *if they are inside the* 2nd *type (or the* 4th *type) critical sub-polygon: p always starts by zig-zag guard sub-state. During zig-zag guard, the pursuer might retreat beyond* p_{ref} *in which case the pursuer switches to the* horizontal guard *sub-state, similar to the vertical guard, in order to recover its progress.*

2. *If they are inside the* 3rd *type critical sub-polygon: the pursuer starts by zig-zag guard if e has appeared inside the first quadrant of p. Otherwise, if e has appeared inside the fourth quadrant, the pursuer starts by simple guard. Similar to the previous case, during zig-zag guard or simple guard, the pursuer might retreat beyond* p_{ref} *in which case it switches to the vertical guard sub-state.*

Algorithm 2: Guard Strategy

```
Input Configuration: The evader is visible while $x(v) \leq x(p) < x(e)$ ($v$ is the pocket vertex).
Exit Configuration : One of the following two configurations: (1) pursuer is on $\pi(O_L, e)$
                     while $p \in h[p_{\text{ref}}]$ and $e \in h(p_{\text{ref}})$, or (2) evader disappears inside $h(p_{\text{ref}})$.
The sub-states     : The *Zig-zag Guard* sub-state, the *Simple Guard* sub-state, and the
                     *Vertical Guard* sub-state. Also the *horizontal guard* sub-state when the
                     pursuer is inside the $2^{nd}$ or the $4^{th}$ type critical sub-polygon.
1  if $y(p) > y(e)$ then
2      $state \leftarrow ZigZagGuard$;
3      $l_v = \vec{Y}_{p_{\text{ref}}}$;
4      do *Zig-zag Guard* strategy ;
5      /* The zigzag guard ends up in one of the following configurations: (1) The
          pursuer is on $\pi(O_L, e)$ while $p \in h[p_{\text{ref}}]$ and $e \in h(p_{\text{ref}})$, (2) The evader
          disappears inside $h(p_{\text{ref}})$, or (3) $x(p) = x(e) = x(p_{\text{ref}})$, $y(e) < y(p) \leq y(p_{\text{ref}})$,
          and $e$ is crossing $l_v$ to the left.                                          */
6  else
7      $state \leftarrow SimpleGuard$;
8      define $p_{\text{aux}} \in h[p_{\text{ref}}]$ as explained in Section 4.6.2;
9      $l_v = \vec{Y}_{p_{\text{aux}}}$;
10     do *Simple Guard* strategy ;
11     /* The simple guard ends up in one of the following configurations: (1) $p$ is
          on $\pi(O_L, e)$ while $p \in h[p_{\text{aux}}]$ and $e \in h(p_{\text{aux}})$, or (2) $e$ disappears inside
          $h(p_{\text{aux}})$, or (3) $x(p) = x(e) = x(p_{\text{aux}})$, $y(e) < y(p) \leq y(p_{\text{aux}})$, and $e$ is crossing $l_v$
          to the left.                                                                    */
   /* next state after $p$ exits the zig-zag guard state or the simple guard state.   */
12 if $p$ is on $\pi(O_L, e)$ then
13     do *Extended Lion's Move* strategy
14 else if $e$ has disappeared then
       /* update $p_{\text{ref}}$                                                          */
15     Let $v'$ be the vertex that $e$ has disappeared behind;
16     if $state = ZigZagGuard$ and $v' \in Chain_U$ then
17         $p_{\text{ref}} \leftarrow v'$;
18     else if $state = SimpleGuard$ and $v' \in Chain_U$ then
19         $p_{\text{ref}} \leftarrow v'$;
20     else if $state = SimpleGuard$ and $v' \in Chain_L$ then
21         $p_{\text{ref}} \leftarrow p_{\text{aux}}$;
22     do *Search* strategy
23 else if $x(p) = x(e)$ and $e$ is crossing $l_v$ to the left then
24     do Vertical Guard strategy;
25     /* The vertical guard ends up in one of the following two configurations:
          (1) The pursuer is on $\pi(O_L, e)$ while $p \in h[p_{\text{ref}}]$ and $e \in h(p_{\text{ref}})$, (2) The evader
          disappears inside $h(p_{\text{ref}})$.                                          */
```

4.6.1 Zig-Zag Guard

After finding the evader, p switches to the zig-zag guard sub-state if e is inside the fourth quadrant of p. The goal of this state is to establish the extended lion's move state while the invariants and the notions of progress are maintained. The *Zig-Zag Guard* strategy is shown in Algorithm 3.

Algorithm 3: Zig-zag Guard Strategy

Input Configuration: The evader is visible while $x(v) \leq x(p) < x(e)$ and $y(p) > y(e)$. Here, v is the pocket vertex.

Exit Configuration : One of the following three configurations: (1) The pursuer is on $\pi(O_L, e)$ while $p \in h[p_{\text{ref}}]$ and $e \in h(p_{\text{ref}})$, (2) The evader disappears inside $h(p_{\text{ref}})$, or (3) $x(p) = x(e) = x(p_{\text{ref}})$, $y(e) < y(p) \leq y(p_{\text{ref}})$, and e is crossing $l_v = \vec{Y}_{p_{\text{ref}}}$ to the left.

```
1 repeat
2     if $x(p) < x(e)$ then
          if $p$ is below $\pi(O_L, e)$ then
3             move in the positive $\vec{y}$ direction
          else if $p$ is above $\pi(O_L, e)$ then
4             move in the negative $\vec{y}$ direction
      else
5         move in the negative $\vec{x}$ direction;
      end
  until (1) The pursuer catches up with $\pi(O_L, e)$, (2) The evader disappears, or (3)
  $x(p) = x(e) = x(p_{\text{ref}})$, $y(e) < y(p) \leq y(p_{\text{ref}})$, and $e$ is crossing $l_v = \vec{Y}_{p_{\text{ref}}}$ to the left;
```

In order to establish the extended lion's move state, the pursuer moves toward $\pi(O_L, e)$ along the x-axis or the y-axis: if $x(p) < x(e)$, the pursuer moves parallel to the $\vec{y}$-axis. Otherwise, if e moves to a point which is to the left of p, then p moves in the negative $\vec{x}$ direction. We show that by following these zig-zag moves the evader will remain inside the fourth quadrant of p (Lemma 4.6.1). See Fig. 4.8 and Fig. 4.9(a).

Let p_{ref} be the current reference vertex used for tracking progress. According to the invariants, the pursuer must ensure that e is to the right of p_{ref}. However, in the strategy described above, e can force p to move along the negative x-axis direction. Therefore, the evader can re-contaminate the region to the left of p_{ref}. That is it crosses the vertical line passing through p_{ref} to the left, and enters the region $(Q - h[p_{\text{ref}}])$.

This violates the invariant (I1). See Fig. 4.9(a). Also note that this is the third exit configuration in Algorithm 3. At this time, the pursuer switches to the *vertical guard* strategy. The vertical guard strategy (presented in section 4.6.3) ensures that the evader will be pushed back to the right of p_{ref} and hence the invariant and the progress are recovered. In the following lemma we show that the invariants are maintained as p exits the zig-zag guard state. This lemma also presents the progress that p gains at the end of zig-zag guard state.

Lemma 4.6.1 (Zig-zag guard progress)**.** *When the pursuer exits the zig-zag guard sub-state, the players are in one of the following three configurations:*

- *The pursuer is on $\pi(O_L, e)$ while $p \in h[p_{\text{ref}}]$ and $e \in h(p_{\text{ref}})$.*
- *The evader disappears inside $h(p_{\text{ref}})$.*
- *$x(p) = x(e) = x(p_{\text{ref}})$, $y(e) < y(p) \leq y(p_{\text{ref}})$, and e is crossing $l_v = \vec{Y}_{p_{\text{ref}}}$ to the left.*

For each of these configurations, the pursuer achieves the following notions of progress correspondingly:

- *the pursuer switches to the L state while $R(p_{\text{ref}}) \leq R(p) < R(e)$*
- *the pursuer switches to the S state while progress (P1) or (P2) is guaranteed.*
- *the pursuer switches to the* vertical guard *sub-state.*

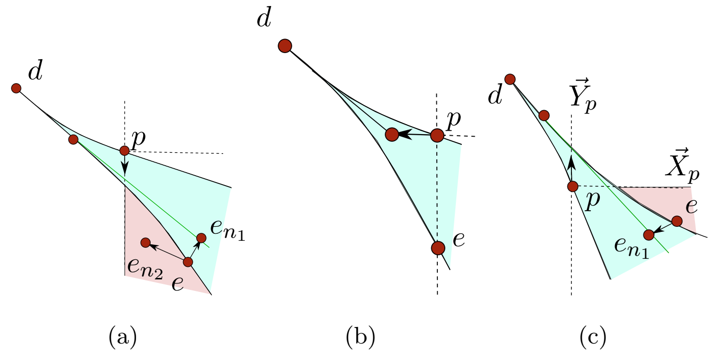


Figure 4.8: The zig-zag guard strategy. (a) When $x(p) < x(e)$ and p is above $\pi(O_L, e)$. (b) When $x(p) = x(e)$ and p is above $\pi(O_L, e)$. (c) When $x(p) < x(e)$ and p is below $\pi(O_L, e)$.

Proof. Let p_0 and e_0 be the positions of p and e at the beginning of the zig-zag guard. Observe that $parent(p_0)$ is in the second quadrant of p_0 (Lemma 4.8.3). Now consider the funnel [76] formed by $\pi(O_L, e)$ and $\pi(O_L, p)$. Let d be the deepest common vertex between these two paths. The shortest path to all points inside this funnel starts by $\pi(O_L, d)$ and then continues inside the funnel. Observe that as e moves, $\pi(O_L, e)$ changes continuously. See Fig. 4.8.

Suppose that p is below $\pi(O_L, e)$ (Fig. 4.8(c)). Then p is getting closer to $\pi(O_L, e)$ *just* by moving in the positive $\vec{y}$ direction. Note that the slope of the edge between p and $parent(p)$ is negative (Lemma 4.8.3). Hence, if the evader tries to cross $\vec{Y}_p$ to the left, p will be on $\pi(O_L, e)$ and thus it can switch to the L state. Therefore, e has to remain inside the fourth quadrant of p until one of the following happens: (1) p is on $\pi(O_L, e)$, (2) the evader disappears. Now, let v be the pocket vertex that the pursuer searched right before this zig-zag guard state. According to Lemma 4.5.3, $x(v) \leq x(p)$. Also, according to the invariant (I2) we have $v \in h[p_{\text{ref}}]$. Thus, $p \in h[p_{\text{ref}}]$ and $e \in h(p_{\text{ref}})$. Hence, invariant (I1) is valid at the end of zig-zag guard state.

Similarly, if the pursuer is above $\pi(O_L, e)$ it moves in the negative $\vec{y}$ direction when $x(p) < x(e)$, and then to the left, i.e. in the negative $\vec{x}$ direction, at the moment that $x(p) = x(e)$. See Fig. 4.8 parts (a) and (b), also Fig. 4.9(a). These zig-zag moves continue until (1) the pursuer is on $\pi(O_L, e)$, (2) the evader disappears, or (3) the players cross $\vec{Y}_{p_{\text{ref}}}$ to the left (Fig. 4.9(b)). At the time that each of these happen, the players are inside $h(p_{\text{ref}})$ and thus invariant (I1) is valid.

Next let us consider invariant (I2) that the pocket vertex is inside $h[p_{\text{ref}}]$. Initially, $p_{\text{ref}} = O_L$. Thus, the invariant holds. According to the above argument, as the game proceeds to zig-zag guard state the invariant is still valid since when e disappears it is inside $h(p_{\text{ref}})$. Therefore, using induction on time we can prove that the pocket vertex is inside $h[p_{\text{ref}}]$.

So far we have shown that the invariants (I1) and (I2) are maintained. Next, let us consider the pursuer's progress. As discussed above, the pursuer finishes the zig-zag guard state when it reaches $\pi(O_L, e)$, or when the evader disappears, or when the players move to the left of p_{ref}. In the first case, since p is on $\pi(O_L, e)$ we have $R(p) < R(e)$. Also, since $p \in h[p_{\text{ref}}]$ we have $R(p_{\text{ref}}) \leq R(p) < R(e)$.

Now consider the case that the evader disappears. Let v' be the new pocket vertex that e disappears behind, and v be the pocket vertex that was used right before this zig-zag guard state. Also, let p_{ref} be the reference vertex at the beginning of this zig-zag guard state, and p'_{ref} be the new reference vertex at the end of this state.

1. $v \in Chain_L$: suppose that at the beginning of the zig-zag guard the pursuer was below $\pi(O_L, e)$. Then, $v' \in h(v)$ since p is only moving upward. If $v' \in Chain_L$ we do not update p_{ref} but we have $v' \in h(v)$ (Progress (P2)). Otherwise, if $v' \in Chain_U$ we set p'_{ref} to v'. Since $p'_{\text{ref}} = v' \in h(v)$ and $v \in h[p_{\text{ref}}]$ (Invariant (I2)), we have $R(p_{\text{ref}}) < R(p'_{\text{ref}})$ (Property 4.3.3) and hence we have Progress (P1).

   Next, suppose that at the beginning of zig-zag guard the pursuer was above $\pi(O_L, e)$. Since e remains inside the fourth quadrant of p, $v \in Chain_L$, and p is moving downward and to the left, e cannot cross v to the left. Hence, v' is also in the fourth quadrant of v. If $v' \in Chain_L$ we do not update p_{ref} but we have $v' \in h(v)$ (Progress (P2)). Otherwise, if $v' \in Chain_U$ we set p'_{ref} to v'. Since $p'_{\text{ref}} \in h(v) \subseteq h(p_{\text{ref}})$ we have Progress (P1).

2. $v \in Chain_U$: then $p_{\text{ref}} = v$. As we showed above, whether p is initially below or above $\pi(O_L, e)$, the evader disappears inside $h(p_{\text{ref}})$. Similar to the previous case, if $v' \in Chain_L$ we have $v' \in h(p_{\text{ref}} = v)$ (Progress (P2)), and if $v' \in Chain_U$ we set p'_{ref} to v' and we have Progress (P1).

In summary, if e disappears during zig-zag guard state, the pursuer updates p_{ref} to v' when $v' \in Chain_U$ and achieves progress (P1) (line 17 in Algorithm 2). Otherwise, if $v' \in Chain_L$ the pursuer achieves progress (P2). $\square$

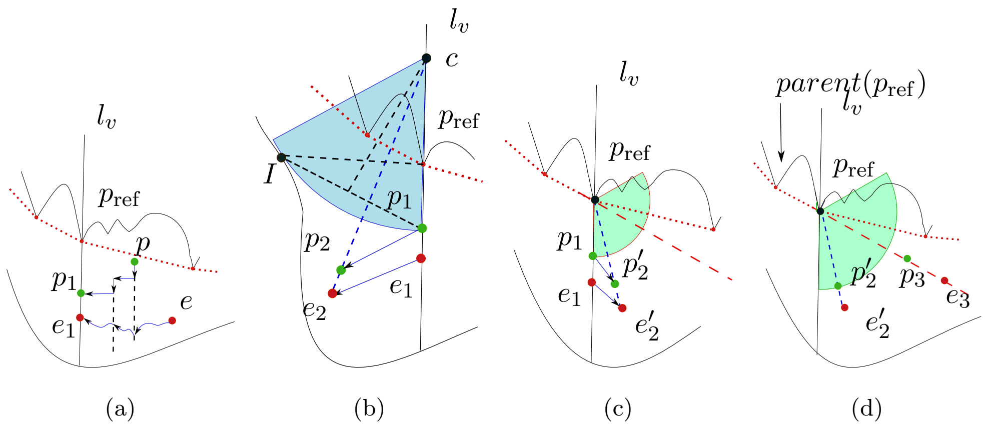


Figure 4.9: The zig-zag guard followed by the vertical guard . The path Π is shown in dots. (a) The zig-zag moves. (b) If e moves to the left of p_{ref}, the pursuer follows it by lion's move with respect to c. (c) Afterwards, if e moves to the right of p_{ref}, the pursuer performs the lion's move with respect to p_{ref}. (d) Finally, p will be on $\pi(O_L, e)$ inside $h(p_{\text{ref}})$.

4.6.2 Simple Guard

After finding the evader, p switches to the simple guard sub-state if e is inside the first quadrant of p. The main goal of the pursuer is to reach $\pi(O_L, e)$ so it can start the extended lion's move state. Since the evader can disappear in the meantime, this translates to establishing the next state which could be the search state or the extended lion's move state while the invariants and the notions of progress are maintained. The *Simple Guard* strategy is presented in Algorithm 4.

The general idea for the pursuer's strategy is the following. See Fig. 4.10, and let p_{ref} be the current reference vertex used for tracking progress. Let v be the vertex that defines the contaminated pocket right before this guard state. When the pursuer starts the simple guard state we have $x(v) \leq x(p) < x(e)$ (the exit configuration of the search state). Therefore at this time, p is to the right of p_{ref} i.e. inside $h[p_{\mathrm{ref}}]$ since $v \in h[p_{\mathrm{ref}}]$ (Invariant (I2)). Now let p_0 be the location of the pursuer at the beginning of the simple guard state. The pursuer moves back toward the vertex p_{ref} along the line segment that connects p_0 to p_{ref} (p_1 in Fig. 4.10(a)). It continues moving toward p_{ref} until it reaches p_{ref}, or e crosses the ray shot from p in the direction of p_{ref} to p (p_2 and e_2 in Fig. 4.10(b)). In both of these cases, p follows e by lion's move with respect to the center p_{ref}. The pursuer continues with this lion's move until one of the following configurations hold: 1) either p reaches $\pi(O_L, e)$ while it is inside $h(p_{\mathrm{ref}})$, 2) or e moves to the region which is to the left of p_{ref} i.e. it crosses the vertical line which passes through p_{ref} to the left and re-contaminates $h(p_{\mathrm{ref}})$. See p_3 and e_3 in Fig. 4.10(b).

In the former case, when the pursuer catches up with $\pi(O_L, p)$, it switches to the extended lion's move state, and since it is inside $h(p_{\mathrm{ref}})$ the invariant (I1) is maintained. In the latter case, when e moves to the left of p_{ref}, invariant (I1) is violated. At this time, the pursuer switches to the next state which is called the vertical guard state in order to push the evader back to the right of p_{ref}, and re-establish invariant (I1). The vertical guard strategy, presented in section 4.6.3, guarantees that the pursuer reaches $\pi(O_L, e)$ inside $h(p_{\mathrm{ref}})$ and therefore is ready for the extended lion's move state while invariant (I1) holds.

The simple guard strategy described above has two subtle points. First, p_{ref} must be visible to p_0 so that p can move along the segment that connects p_{ref} to p_0. However, it might be the case that p_{ref} is not visible to p_0 since p_0 is the location of the pursuer at the time that p has found e. Therefore, we define an auxiliary vertex, referred to as p_{aux}, so that it is visible to p_0 and moreover $p_{\mathrm{aux}} \in h[p_{\mathrm{ref}}]$. The pursuer performs the aforementioned strategy with respect to p_{aux} i.e. it moves toward p_{aux} and the rest of the strategy. Since the simple guard strategy ensures that in the next state the players are inside $h(p_{\mathrm{aux}})$, and because $p_{\mathrm{aux}} \in h[p_{\mathrm{ref}}]$ the invariant (I1) is maintained.

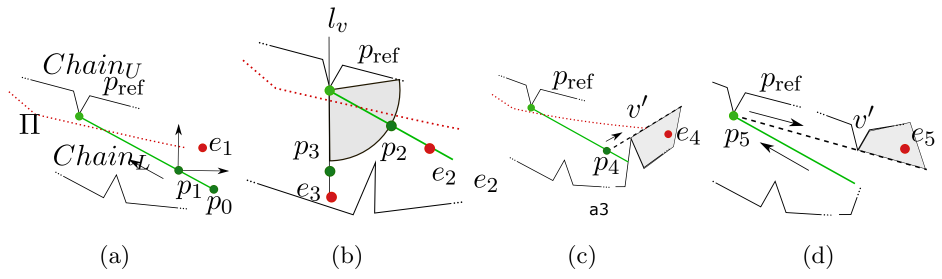


Figure 4.10: The Simple Guard. (a) p moves toward p_{ref}. (b) As e crosses the ray from p to p_0, p performs lion's move w.r.t. p_{ref} (p_2 and e_2). If during this lion's move e enters the region to the left of p_{ref}, the pursuer switches to the vertical guard sub-state e.g. p_3 and e_3. (c) If e hides inside $pocket(v', \vec{r'})$, p moves toward v' if $v' \in Chain_L$, (d) otherwise if $v' \in Chain_U$ it continues moving toward p_{ref} and then from there it moves toward v'. Note that here, as the pursuer keeps moving back to p_{ref}, the new pocket formed by the ray connecting p to v', includes the initial hiding pocket $pocket(v', \vec{r'})$. Also in this example $p_{\text{aux}} = p_{\text{ref}}$.

The second important part of the strategy is that p_0 must be closer to all points on the segment between p_0 and p_{aux} than the evader so that p can prevent e from crossing this segment and thus escaping to the cleared region $(Q - h[p_{\text{ref}}])$. Notice that if the pursuer does not prevent this type of re-contamination the evader will be above the pursuer i.e. $y(p) < y(e)$. Therefore p cannot force e back to $h(p_{\text{ref}})$ by performing the vertical guard strategy as it does when $y(p) > y(e)$ (we will see in section 4.6.3 that one of the conditions that p is allowed to perform the vertical guard sub-state is $y(p) > y(e)$). Instead the pursuer prevents this situation by guaranteeing that it is closer to all points on the segment between p_0 and p_{aux}. This, in addition to the capture condition that e will be captured if its distance to p is less than one unit, ensures that e will be captured if it tries to cross the segment between p_{aux} and p. Finally, the pursuer ensures that it is closer to p_0 by guaranteeing that the angle between the aforementioned segment and the x-axis is less than or equal to α (Lemma 4.8.6).

An illustrative example of the auxiliary vertex p_{aux} is shown in Fig. 4.11(a). The interested reader is referred to the Appendix, Section 4.8.3, for the definition of the auxiliary vertex p_{aux} based on the structure of the polygon and the location of the pursuer.

Lemma 4.6.2 (Simple guard progress)**.** *When the pursuer exits the simple guard sub-state, the players are in one of the following three configurations:*

- *The pursuer is on $\pi(O_L, e)$ while $p \in h[p_{\text{aux}}]$ and $e \in h(p_{\text{aux}})$.*
- *The evader disappears inside $h(p_{\text{aux}})$.*
- *$x(p) = x(e) = x(p_{\text{aux}})$, $y(e) < y(p) \leq y(p_{\text{aux}})$, and e is crossing $l_v = \vec{Y}_{p_{\text{aux}}}$ to the left.*

For each of these configurations, the pursuer achieves the following notions of progress correspondingly:

- *the pursuer switches to the L state while $R(p_{\text{ref}}) \leq R(p_{\text{aux}}) \leq R(p) < R(e)$*
- *the pursuer switches to the S state while progress (P1) or (P2) is guaranteed.*
- *the pursuer switches to the* vertical guard *sub-state.*

Proof. From the description above, the exit configuration of simple guard is one of the following: (1) the L state inside $h(p_{\text{aux}})$, (2) the S state inside $h(p_{\text{aux}})$, or (3) vertical guard while the player are crossing l_v to the left. In case of the L state, since $p_{\text{aux}} \in h[p_{\text{ref}}]$ we would have $R(p_{\text{ref}}) \leq R(p_{\text{aux}}) \leq R(p) < R(e)$.

In case of the S state, let v' be the new pocket vertex. Then $v' \in h(p_{\text{aux}}) \subseteq h(p_{\text{ref}})$. Therefore, the invariant (I2) is valid.

Next, let us consider the progress. Let p'_{ref} denote the new reference vertex that is updated in this guard state. If $v' \in Chain_U$ we set p'_{ref} to v'. Since $p'_{\text{ref}} = v' \in h(p_{\text{ref}})$ we have $R(p_{\text{ref}}) < R(p'_{\text{ref}})$ (Property 4.3.3) and hence we have Progress (P1).

If $v' \in Chain_L$ we update p_{ref} to p_{aux}. Note that p_{aux} can be the same as p_{ref}. However, we show that $v' \in h(v)$. Suppose that $v \in Chain_L$ and p_{aux} is to the left of v. The remaining situations are similar. Now if $v' \notin h(v)$, the pocket $pocket(v', \vec{r'})$ defined by v' will be a simple pocket and thus by performing the simple pocket strategy (section 4.8.5) the pursuer can force the evader to exit $pocket(v', \vec{r'})$ and continue the simple pocket strategy. See Fig. 4.13(a) for an illustration. □

The complete description of simple guard strategy is given in Algorithm 4. Notice that when the evader disappears while p is moving back toward p_{ref}, the pursuer's reaction depends on whether the hiding vertex v' is from $Chain_L$ or $Chain_U$ (lines 12-16 in Algorithm 4). An example is shown in Fig. 4.10(c) parts (c) and (d). When $v' \in Chain_L$, the pursuer moves toward v' until it reaches v' at which time it switches to the S state (p_4 and e_4 in Fig. 4.10(c)). When $v' \in Chain_U$, the pursuer continues moving back toward p_{ref} and if in the meantime e crosses $\overrightarrow{ray}$ (line 3) the pursuer performs the same strategy from line 5.3.6. In the following, we briefly explain the reason that p must make distinction between $v' \in Chain_L$ and $v' \in Chain_U$:

- $v' \in Chain_L$: in this case, if the pursuer keeps moving back toward p_{ref}, it cannot keep track of the hiding vertex v'. For example, in Fig. 4.11(c), at the time that e disappears, p defines the hiding pocket with respect to $v' = v_1$. If p continues moving back toward p_{ref}, at p_2 the hiding pocket with respect to v_1 doesn't include e (Fig. 4.11(d)). In other words, p cannot keep track of the hiding vertex v'.

- $v' \in Chain_U$: in this case $x(v')$ can be less than $x(p)$. Therefore, if the pursuer moves toward v' the evader *can cross* the segment between p_{ref} and p (Fig. 4.11(b)) and thus it escapes to the previously cleared region.

Algorithm 4: Simple Guard Strategy

Input Configuration: The evader is visible while $x(v) \leq x(p) < x(e)$ and $y(p) < y(e)$. Here, v is the pocket vertex.

Exit Configuration : One of the following three configurations: (1) The pursuer is on $\pi(O_L, e)$ while $p \in h[p_{\text{aux}}]$ and $e \in h(p_{\text{aux}})$, (2) The evader disappears inside $h(p_{\text{aux}})$, or (3) $x(p) = x(e) = x(p_{\text{aux}})$, $y(e) < y(p) \leq y(p_{\text{aux}})$, and e is crossing $l_v = \vec{Y}_{p_{\text{aux}}}$ to the left.

```
1  define $p_{\text{aux}}$ as explained in section 4.6.2;
2  let $p_0$ be the location of $e$ at the beginning of the simple guard;
3  let $\overrightarrow{ray}$ be the ray shot from $p$ in the direction of $p_{\text{aux}}$ to $p_0$;
4  **repeat**
5      move toward $p_{\text{aux}}$ along the segment $p_0 p_{\text{aux}}$;
   **until** *$e$ crosses $\overrightarrow{ray}$, or $e$ disappears, or $p$ reaches* $p_{\text{aux}}$;
6  **if** *$e$ has crossed $\overrightarrow{ray}$, or $p$ has reached* $p_{\text{aux}}$ **then**
7      **if** *$e$ has crossed $\overrightarrow{ray}$, or $p$ has reached* $p_{\text{aux}}$ *and $e$ is visible* **then**
8          **repeat**
9              do lion's move with respect to the center $p_{\text{aux}}$;
           **until** *$p$ is on $\pi(O_L, e)$, or $e$ disappears, or $x(p) = x(e) = x(p_{\text{aux}})$ and $e$ is
           crossing $l_v = \vec{Y}_{p_{\text{aux}}}$ to the left*;
10     **else if** *$p$ has reached* $p_{\text{aux}}$ *and $e$ is hidden behind $v'$ inside pocket$(v', \vec{r'})$* **then**
11         move toward $v'$;
       **end**
12 **else if** *$e$ has disappeared behind $v'$ inside pocket$(v', \vec{r'})$* **then**
       /* $p$ is not at $p_{\text{aux}}$ yet.                                              */
13     **if** $v' \in Chain_L$ **then**
14         move toward $v'$ until $p$ is at $v'$;
15     **else if** $v' \in Chain_U$ **then**
16         continue moving toward $p_{\text{aux}}$ and the rest of the strategy at line 5.3.6;
17         in the meantime keep track of the hiding vertex $v'$;
       **end**
   **end**
```

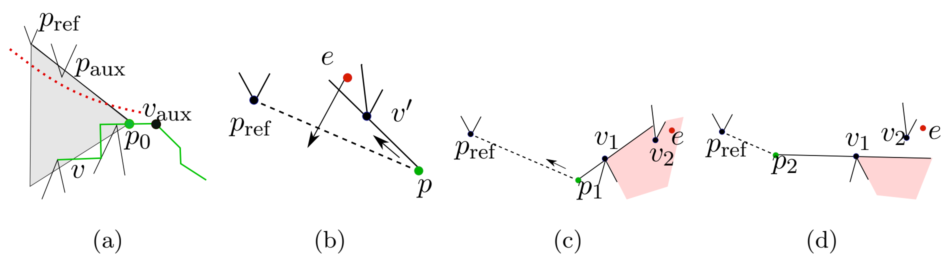


Figure 4.11: (a) An example of p_{aux}. (b), (c) and (d) Examples when e disappears behind a vertex, namely v', during simple guard. The pursuer has to make distinction between $v' \in Chain_U$ and $v' \in Chain_L$, otherwise e can escape. Refer to the text.

4.6.3 Vertical Guard

The vertical guard strategy is presented in Algorithm 5. The pursuer switches to the vertical guard sub-state from either the zig-zag guard sub-state or the simple guard sub-state. See Fig. 4.3. Let p_{ref} be the current reference vertex used for tracking the progress. The condition for state transition to the vertical guard sub-state is when the evader re-contaminates the region to the left of p_{ref} i.e. it enters the region $P - h[p_{\text{ref}}]$ and violates the invariant (I1). Since $h(p_{\text{ref}})$ is defined as the set of points to the right of p_{ref}[8], this condition is in fact when $x(p) = x(e) = x(p_{\text{ref}})$, $y(e) < y(p) \leq y(p_{\text{ref}})$ and e is moving to the left of p_{ref}. Let l_v denote the line $\vec{Y}_{p_{\text{ref}}}$. Then, the pursuer switches to the vertical guard state when e crosses l_v to the left. The goal of the vertical guard strategy is to push the evader back to the right of p_{ref} and hence re-establish the invariant (I1).

The vertical guard strategy is composed of two parts: lion's move with respect to a center c (which we define soon) and lion's move with respect to the center p_{ref}. The pursuer uses c as the center for the lion's move if e crosses l_v to the left. It also uses p_{ref} as the center for lion's move if e crosses l_v to the right. The role of the circle centered at c is to push e back to the right of p_{ref} (inside $h(p_{\text{ref}})$) and re-establish the invariant (I1). The role of the other circle centered at p_{ref} is to force e to cross the ray connecting $parent(p_{\text{ref}})$ to p_{ref} and hence to establish the extended lion's move state. See Fig. 4.9. The vertical guard strategy is as follows, see Fig. 4.9:

[8] Inside the first type critical sub-polygon.

1. As the evader crosses the vertical line l_v to the left, p follows him by lion's move with respect to c, see Fig. 4.9-(b) p_1 and e_1 to p_2 and e_2 respectively.

   If the evader disappears behind the lower chain vertices which are to the left of l_v, the pocket would be of a special form that we call it *simple pocket*[9]. In simple pockets the pursuer has a relatively simple strategy so that the evader *has to* exit the pocket to prevent capture. See section 4.8.5 for definition of simple pocket and the corresponding pursuit strategy. Therefore, when e disappears somewhere to the left of l_v, the pursuer can repel him outside the hiding pocket by simple pocket strategy.

   Consequently, as long as e is on the left side of l_v, the distance cp increases while p lies on ce. As a result, e will be pushed to the right of l_v after finite time (Fig. 4.9-(c) where e is at e'_2).

2. As the evader crosses the vertical line l_v to the right, p switches to lion's move with respect to the center p_{ref} (Fig. 4.9-(c) where p moves to p'_2).

3. This back and forth switch between lion's move with respect to centers c and p_{ref} continues until one of the following two configurations hold: 1) the evader disappears behind a vertex to the right of p_{ref}, 2) or the extended lion's move is established (p_3 and e_3 in Fig. 4.9-(d)).

[9] See Lemma 4.8.11.

Algorithm 5: Vertical Guard Strategy

Input Configuration: The evader and the pursuer are both on the line l_v and e is crossing l_v to the left. In other words, $x(p) = x(e) = x(c_{\text{aux}})$ and $y(e) < y(p) \leq y(c_{\text{aux}})$, and e is moving to the left of the vertex c_{aux}. Refer to the lines 1- 4 for definition of the vertex c_{aux} and the line l_v.

Exit Configuration : Either (1) the pursuer is on $\pi(O_L, e)$ while $p \in h[p_{\text{ref}}]$ and $e \in h(p_{\text{ref}})$, or (2) the evader disappears inside $h(p_{\text{ref}})$.

```
1  if the previous state is zig-zag guard then
2      $l_v = \vec{Y}_{p_{\text{ref}}}$;   $c_{\text{aux}} \leftarrow p_{\text{ref}}$;
3  else if the previous state is simple guard then
4      $l_v = \vec{Y}_{p_{\text{aux}}}$;   $c_{\text{aux}} \leftarrow p_{\text{aux}}$;
   end
5  $I \leftarrow \vec{X}_{c_{\text{aux}}} \cap \partial Q$ ;
6  $c =$ bisector of $pI \cap l_v$;
7  repeat
8      if $e$ is to the left of $c_{\text{aux}}$ then
9          repeat
10             do lion's move with respect to the center $c$;
           until $e$ disappears somewhere to the left of $c_{\text{aux}}$, or $e$ moves to the right of $c_{\text{aux}}$;
11         if $e$ has disappeared somewhere to the left of $c_{\text{aux}}$ then
12             perform the simple pocket strategy presented in section 4.8.5;
               /* as a result of the simple pocket strategy the evader is forced to
                  exit the hiding pocket while $p$ is on the entrance of the pocket and
                  their distance has been increased.                              */
13             continue from line 9 ;
14         else if $e$ has moved to the right of $c_{\text{aux}}$ then
15             continue from line 16
           end
16     else if $e$ is to the right of $c_{\text{aux}}$ then
17         repeat
18             do lion's move with respect to the center $c_{\text{aux}}$;
           until $p$ is on $\pi(O_L, e)$ to the right of $c_{\text{aux}}$, or $e$ disappears somewhere to the right of
           $c_{\text{aux}}$, or $e$ moves to the left of $c_{\text{aux}}$;
19         if $e$ has moved to the left of $c_{\text{aux}}$ then
20             continue from line 8;
21         else
22             exit the vertical guard sub-state;
           end
   until pursuer catches up with $\pi(O_L, e)$ inside $h(c_{\text{aux}})$, or (2) evader disappears inside $h(c_{\text{aux}})$;
```

Next, let us proceed with the definition of the center c. Let I be the intersection between the horizontal line passing through p_{ref} and ∂P, see Fig. 4.12(a). Then c is the intersection between bisector of pI and the line l_v.

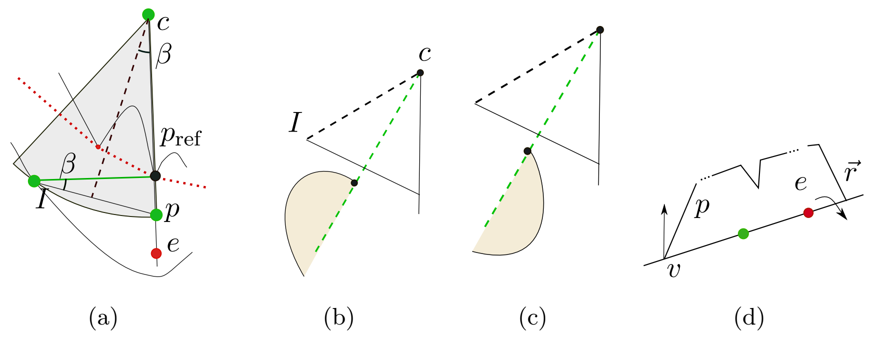


Figure 4.12: (a) the vertical guard (Π is shown in dots). (b) These pockets are impossible. (c) The possible pockets before l_v are simple pockets. (d) The simple pockets.

Remark 6. *When the state before the vertical guard state is the simple guard state, the pursuer defines the center c based on p_{aux} instead of p_{ref}. In Algorithm 5, we use the notation c_{aux} to refer to p_{ref} or p_{aux}. See lines 1-4 in Algorithm 5.*

The lion's move circle centered at c has the following important properties. First, all upper chain vertices before p_{ref} are above $p_{\text{ref}}I$ and thus the lion's move with respect to c is feasible (Lemma 4.8.13).

Second, the lion's move circle centered at c prevents e from hiding behind the upper chain vertices which are to the left of p_{ref} without being captured (Lemma 4.8.10). Also see Fig. 4.6-(e). These vertices are the ones on $Chain_U$ with their x-coordinate less than p_{ref}. In addition, the circle is defined such that if e disappears behind lower chain vertices which are to the left of p_{ref}, the resulting pocket would be a simple pocket (Lemma 4.8.11), pockets that p has a relatively simple strategy to push e out of the pocket (section 4.8.5).

Third, the initial radius of this circle is upper bounded which is necessary for the lion's move with respect to center c to result in progress in finite number of steps [77]. In the following, we show that the angle α plays an important role in bounding the radius. Let r be the radius of the lion's circle centered at c i.e. $r = cp$.

Lemma 4.6.3. *The initial radius (r) of the circle defined above is upper bounded by $r \leq \frac{2l^2}{h_{min}}$ where h_{min} is a lower bound for $h = y(p_{\text{ref}}) - y(p)$ at the beginning of vertical guard ($h_{min} \leq h$).*

Proof. Let β be the angle between pI and the horizontal line passing from c_{aux}, see Fig. 4.12(a). Also let $2l = pI$. We have $\sin\beta = \frac{l}{r} = \frac{h}{2l}$. Hence $r = \frac{2l^2}{h}$. Therefore $r \leq \frac{2l^2}{h_{min}}$. □

In Lemma 4.8.4 and Lemma 4.8.8, we provide $h_{min} = \sin\alpha$ as the lower bound for h at the beginning of the vertical guard strategy.

Corollary 4.6.4. *The initial radius of the vertical guard circle centered at c is upper bounded by $r \leq \frac{D^2}{2\sin\alpha}$. According to Definition 4.5.2, we have r is $O(D^3)$.*

Lemma 4.6.5 (Vertical guard progress)**.** *At the end of vertical guard the pursuer achieves Progress (P1) or (P2).*

Proof. Let $pocket(v, \vec{r})$ be the pocket searched in the previous search state. Recall that if the vertical guard is invoked from zig-zag guard then $c_{\text{aux}} = p_{\text{ref}}$ and if it is invoked from simple guard then $c_{\text{aux}} = p_{\text{aux}}$ (Remark 6).

1. from zig-zag guard ($c_{\text{aux}} = p_{\text{ref}}$):

   (a) $v \in Chain_U$: hence $p_{\text{ref}} = v$. The next state after the vertical guard is either L or S. As the evader exits the vertical guard state the players are inside $h(p_{\text{ref}}) = h(v)$. If the next state is S, let $pocket(v', \vec{r'})$ be the corresponding pocket. Then, $v' \in h(v)$. If $v' \in Chain_U$ we set $p_{\text{ref}} = v'$ and we have Progress (P1). If $v' \in Chain_L$ we have Progress (P2).

   (b) $v \in Chain_L$: this case is not possible. This is because: the evader is inside the fourth quadrant of p (zig-zag guard state), $v \in h[p_{\text{ref}}]$ according to invariant (I2), and the search path used for searching $pocket(v, \vec{r})$ ensures that the evader cannot cross p_{ref} to the left. See Fig. 4.13-(b).

2. from simple guard ($c_{\text{aux}} = p_{\text{aux}}$):

(a) $v \in Chain_U$: then $p_{\text{ref}} = v$ and $p_{\text{aux}} \in h[p_{\text{ref}}]$. At the time that p exits the vertical guard state the players are inside $h(p_{\text{aux}})$ which is a subset of $h(p_{\text{ref}})$. The pursuer achieves Progress (P1) or (P2) similar to the above case.

(b) $v \in Chain_L$: refer to the definition of p_{aux} in section 4.6.2 and note that p_{aux} can be to the left or to the right of v. The case that p_{aux} is to the right of v is similar to the above cases.

Suppose that p_{aux} is to the left of v. Referring to the definition of p_{aux} in section 4.6.2 this is the case only when p_0 is in between v and v_{aux} (the first definition in section 4.6.2). Suppose that during vertical guard (lion's move w.r.t. p_{aux}) the evader disappears behind v'. When $v' \in Chain_U$ we have $v' \in h(p_{\text{aux}}) \subseteq h(p_{\text{ref}})$. We set p_{ref} to v' and achieve progress (P1).

When $v' \in Chain_L$ there are two cases. The first is when $x(v) < x(v')$ which we have progress (P2). The second is when $x(v') \leq x(v)$ in which case the resulting pocket $pocket(v', \vec{r'})$ would be a simple pocket and the pursuer performs the simple pocket strategy in section 4.8.5 in order to resume the lion's move with respect to p_{aux}. See Fig. 4.13-(a).

□

Lemma 4.6.6 (The time spent in the guard state)**.** *The pursuer exits the guard state and switches to the next state in $O(n^4D^{11})$ steps where n is the number of vertices of Q and D is the diameter of Q.*

Proof. Let T_1 be the time spent in simple guard, T_2 be the time spent in vertical guard, and T_3 be the time spent in the zig-zag guard state. Then the guard time is at most $T_1T_2 + T_3T_2$.

The vertical guard strategy is composed of simple pocket strategy (to the left of l_v) and lion's move with respect to c_{aux} or c. In the worst case, every single step of the lion's move can be followed by a round of simple pocket strategy. Therefore, the total time in vertical guard would be the product of the time spent in lion's move and the simple pocket strategy. The time spent in simple pocket strategy is $O(n^2D^3)$ (Lemma 4.8.14). The initial radius of the circle centered at c used during vertical guard is $O(D^3)$ (Corollary 4.6.4). Hence the lion's move with respect to c during vertical guard takes $O(nD^6)$ steps [75]. Therefore, the vertical guard state takes $T_2 = O((nD^6) \cdot (n^2D^3)) = O(n^3D^9)$.

The simple guard is lion's move with respect to p_{aux} which can take at most $T_1 = O(nD^2)$. The zig-zag guard is composed of zig-zag moves which are of time $T_3 = O(D)$.

Thus, the guard time is $O((nD^2) \cdot (n^3D^9) + D \cdot (n^3D^9)) = O(n^4D^{11})$. □

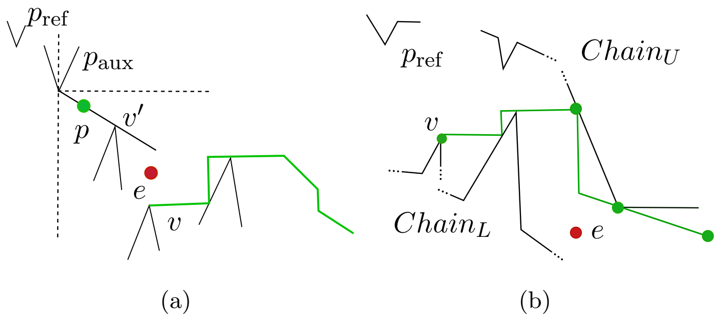


Figure 4.13: (a) When $v \in Chain_L$ and e disappears behind $v' \in Chain_L$ so that $x(v') < x(v)$, the resulting pocket is a simple pocket. (b) When $v \in Chain_L$ and e appears in the fourth quadrant, e cannot cross p_{ref} to the left since it is confined with ∂Q.

4.6.4 Horizontal Guard

We now present the *horizontal guard* strategy which is the counterpart of vertical guard in the second type critical sub-polygon. Suppose that the horizontal guard has been invoked in the 2^{nd} type. Recall that at this time $x(v) \leq x(p) < x(e)$ and $y(p) = y(e) = y(v)$ (Lemma 4.6.1). The center c for the horizontal guard is found as follows: Let I be the intersection between ∂Q and $\vec{Y_v}$. Then c is the intersection point between bisector of pI and $\vec{X_v}$. See Fig. 4.22(b). Symmetric to what we saw in vertical guard, the pursuer performs lion's move with respect to c or v as the evader moves below $l_h = \vec{X_v}$ or above $l_h = \vec{X_v}$. This continues until the next state is established in $h(v)$.

4.7 Analysis of Capture Time

We are now ready to present the proof of Theorem 4.3.4 which gives the worst capture time of the proposed pursuit strategy.

Proof of Theorem 4.3.4. Suppose p is currently in a combined (SG) state. In Lemma 4.6.1, Lemma 4.6.2, and Lemma 4.6.5, we showed that after finite time this combined state will terminate to another combined (SG) state or an L state.

In the latter case, the aforementioned lemmas ensure that $R(p_{\text{ref}}) \leq R(p) < R(e)$. Moreover p is on $\pi(O_L, e)$ and $d(O_L, p)$ is increasing after each step of the L state [75]. Hence either p captures e in the L state or it switches to another (SG) state.

Now consider two consecutive (SG) states and suppose that e is not captured yet. According to the aforementioned lemmas, p achieves progress (P1) or (P2). Since in (P2), v and v' are vertices of Q, after at most n progress updates of type (P2), there would be one progress update of type (P1). Also since p_{ref} is a vertex, at some point $D \leq R(p_{\text{ref}})$. Since D is the diameter of the polygon, at some point $D = R(p_{\text{ref}})$. According to invariant (I1), we must have $D = R(p_{\text{ref}}) < R(e)$. This is a contradiction since $R(e)$ cannot be greater than the diameter.

Next let us provide an upper bound for the number of time-steps required for capture. Let T_1 be the time spent in the guard state plus the time spent in the search state (the time spent in the combined (SG) state). Also let T_2 be the number of steps for a pursuer, which is performing the extended lion's move, to travel the diameter of the polygon. Thus, the number of time-steps between two consecutive combined states (SG) would be $T_1 T_2$. Since $p_{\text{ref}} \in Q$ and $v \in Q$, and we achieve (P1) or (P2) after each (SG) combined state, the total capture time T would be $T = n \cdot nT_1T_2$. According to [75] we have $T_2 = nD^2$. Next, the search time is $O(nD)$ (Lemma 4.5.3), and the guard time is bounded by $O(n^4D^{11})$ (Lemma 4.6.6). Hence $T_1 = n^4D^{11}$ and $T = O(n^2 \cdot (n^4D^{11}) \cdot (nD^2)) = O(n^7D^{13})$. □

4.8 Correctness Proofs

In this section we present the proofs of the lemmas that we saw throughout the chapter. We first go over the properties of monotone polygons. We then prove the correctness of different sub-states of guard strategy.

4.8.1 Properties of Monotone Polygons

Proof of Property 4.3.3. We present the proof for the 4^{th} type critical sub-polygons. The proof for other types is similar. Let $v = v_{i-1}$, see Fig. 4.5 and Fig. 4.14. For other possible v the proof is similar.

According to Lemma 4.8.1 the shortest path to all points to the right of $\overrightarrow{v_{i-2}v_{i-1}}$ passes from v_{i-1}. Hence the length of their shortest path is greater than $d(O_L, v_{i-1})$. Next consider the region in between $\vec{X}_{v_{i-1}}$ and $\overrightarrow{v_{i-2}v_{i-1}}$ and let p be a point in this region. As a corollary of Lemma 4.8.1 we observe that there is a vertex $v_c \in \Pi(v_{m-1}, (v_{i-1})$ that p is the region defined by two rays $\overrightarrow{v_cv_{c+1}}$ and $\overrightarrow{v_{c-1}v_c}$. Moreover p is a descendant of v_c. Next consider the line v_cp and its intersection point with $\vec{X}_{v_{i-1}}$ namely I_p. In the following we will show that $d(v_c, v_{i-1}) < v_cI_p$. Since $v_cI_p < v_cp \leq d(v_c, p)$ we would have $d(v_c, v_{i-1}) < d(v_c, p)$ and thus $d(O_L, v_{i-1}) < d(O_L, p)$.

Let us now present our proof for $d(v_c, v_{i-1}) < v_cI_p$, see Fig. 4.14 top-right. Consider the rays shot in direction of $v_{c'}v_{c'+1}$ where $c \leq c' \leq (i-3)$ and denote their intersection point with $\vec{X}_{v_{i-1}}$ by $I_{c'}$. By induction we show that $v_{c'}I_{c'} + d(v_c, v_{c'}) = d(v_c, I_{c'}) < v_cI_p$.

For the base case consider $c' = c$. Since the angle $v_cI_cI_p$ is greater than 90 degrees we would have $v_cI_c < v_cI_p$.

For the inductive hypothesis let us suppose that the statement is true for c' and prove it for $(c'+1)$. According to our hypothesis we have $v_{c'}I_{c'} + d(v_c, v_{c'}) = d(v_c, I_{c'}) < v_cI_p$. Since the angle $v_{c'+1}I_{c'+1}I_{c'}$ is greater than 90 degrees we would have $v_{c'+1}I_{c'+1} < v_{c'+1}I_{c'}$. We also have $v_{c'}I_{c'} = v_{c'}v_{c'+1} + v_{c'+1}I_{c'}$.

Hence $d(v_c, I_{c'}) = v_{c'}I_{c'} + d(v_c, v_{c'}) = v_{c'}v_{c'+1} + v_{c'+1}I_{c'} + d(v_c, v_{c'}) = d(v_c, v_{c'+1}) + v_{c'+1}I_{c'}$. Since $v_{c'+1}I_{c'+1} < v_{c'+1}I_{c'}$ we would have $d(v_c, v_{c'+1}) + v_{c'+1}I_{c'+1} < d(v_c, I_{c'})$. Recall that $d(v_c, I_{c'}) < v_cI_p$. Thus $d(v_c, I_{c'+1}) < v_cI_p$. □

Lemma 4.8.1. *Let (v_{e-1}, v_e) be an edge on Π. Consider the ray shot in direction of $\vec{r} = \overrightarrow{v_{e-1}v_e}$. Then the shortest path to all vertices to the right of $\vec{r}$ passes through the vertex v_e. See Fig. 4.14 left and middle.*

Proof. For the sake of contradiction suppose that there are some edges on Π that this property does not hold for them. Among these edges let (v_{e-1}, v_e) be the first one i.e.

with smallest v_e. Thus there is a point to the right of $\vec{r} = \overrightarrow{v_{e-1}v_e}$ that the shortest path to that point does not pass from v_e. Then the direct parent of one of its ancestors should be a vertex to the left of $\vec{r}$. Let v be this ancestor and let $v' = parent(v)$. Let $(v_{e'-1}, v_{e'})$ be the edge on Π that the shortest path to v' passes from $v_{e'}$. Since v' is to the left of $\overrightarrow{v_{e-1}v_e}$ and since the property holds for all edges before (v_{e-1}, v_e) we would have $e' \leq (e-1)$. Next observe that $\overrightarrow{v_{e-1}v_e}$ intersects with $\pi(e', v')$. This can be seen by enumeration over all possible situations arisen depending on the type of the critical sub-polygon that (v_{e-1}, v_e) belongs to. For example the case where this edge is in the second type critical sub-polygon and before the summit vertex is depicted in Fig. 4.14 left and middle. In this case since slope of v_{e-1}, v_e is negative and slope of Π we observe that $\overrightarrow{v_{e-1}v_e}$ intersects with $\pi(e', v')$. Next let I_1 and I_2 be the intersection of $\overrightarrow{v_{e-1}v_e}$ with $v'v$ and $\pi(e', v')$ respectively. Then according to triangle inequality we have: $I_1I_2 < v'I_1 + d(v', I_2)$. Hence $vI_1, I_1I_2, \pi(e'I_2), \pi(O_L, e')$ is a shorter path than $\pi(O_L, v) = vI_1, v'I_1, v'I_2, I_2e', \pi(O_L, e')$ which is a contradiction. $\square$

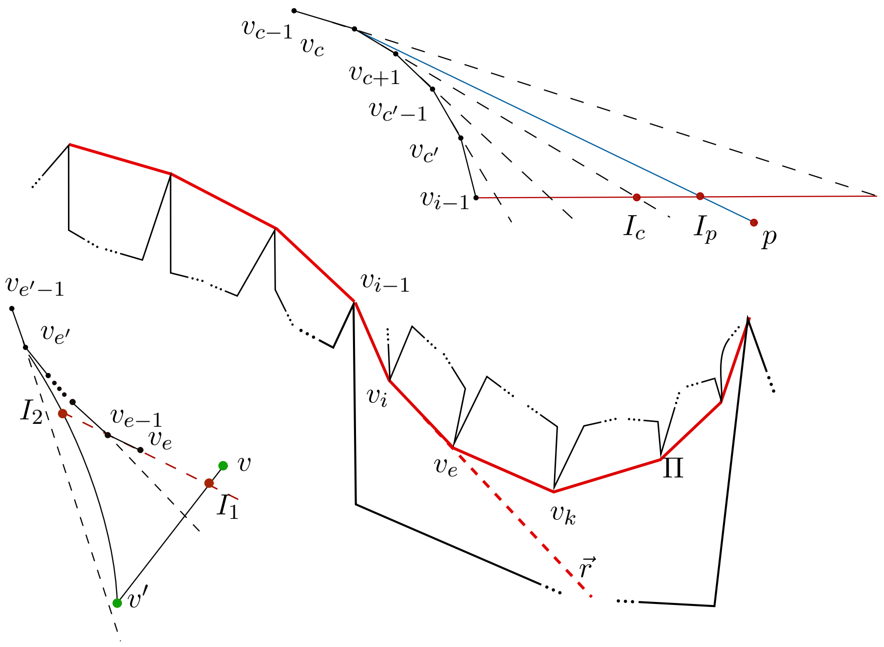


Figure 4.14: proof of Lemma 4.8.1 and Property 4.3.3.

Lemma 4.8.2. *Suppose $v_{i-1}v_i$ and $v_{j-1}v_j$ are two consecutive critical edges and consider $\pi(v_{i-1}, v_{j-1})$ which is also part of Π. Then (see Fig. 4.5-left):*

(i) If $v_i \in Chain_U$, then the slope of edges on $\pi(v_{i-1}, v_j)$ is monotonically increasing.

(ii) If $v_i \in Chain_L$, then the slope of edges on $\pi(v_{i-1}, v_j)$ is monotonically decreasing.

Proof. First, consider a segment wz and a point k with $x(z) < x(k)$, see Fig. 4.5:

1. k is below the ray $\overrightarrow{wz}$: then $slope(zk) < slope(wz)$.
2. k is above the ray $\overrightarrow{wz}$: then $slope(zk) > slope(wz)$.

Let w, z and k be three consecutive vertices on $\pi(v_{i-1}, v_j)$. Now, consider the case that $v_i \in Chain_U$. In this case, k *must* be above the ray wz because otherwise Π *cannot* be a shortest path [76]. Similarly, when $v_i \in Chain_L$, k *must* be below the ray wz. Therefore:

(i) when $v_i \in Chain_U$ the slope of edges is monotonically increasing. (ii) and, when $v_i \in Chain_L$ the slope of edges is monotonically decreasing. □

Lemma 4.8.3. *Consider the search path inside the first type critical sub-polygon. Let p be a point on this part of the search path. Then the slope of the edge that connects p to $parent(p)$ is negative.*

Next consider the 2^{nd} type critical sub-polygons and suppose that p is in this portion of the search path. Then the slope of the edge that connects p to $parent(p)$ is positive.

Proof. First observe for all points p, $x(parent(p)) < x(p)$.

- p is in the first type critical sub-polygon: Note that all points in this part are descendants of v_{i-1}, see Lemma 4.8.1. For the sake of contradiction let us assume that $parent(p)$ is in the third quadrant of p, see Fig. 4.15. In the following we will show that there exist a shorter path than $\pi(O_L, p) = \pi(O_L, parent(p)) + parent(p)p$ which is a contradiction. Note that p is on the search path.

  1. p is on the α-path (Fig. 4.15(a)): Let A be the intersection of $\vec{X}_p$ with $\pi(O_L, parent(p))$. Observe that A is visible to p since all upper chain vertices are above the search path. Because of the triangulation inequality, pA

followed by $\pi(O_L, A)$ is a shorter path than $\pi(O_L, p) = \pi(O_L, parent(p)) + parent(p)p$ which is a contradiction.

2. p is on the step-path (Fig. 4.15(b)): Similar to the previous case.

- p is in the second type critical sub-polygon: The same as the previous case. See Fig. 4.15(c). Note that all points in this part are descendants of the summit vertex s, see Lemma 4.8.1. Also, A is visible to p since all lower chain vertices are below the search path. The rest of the proof is the same as above.

□

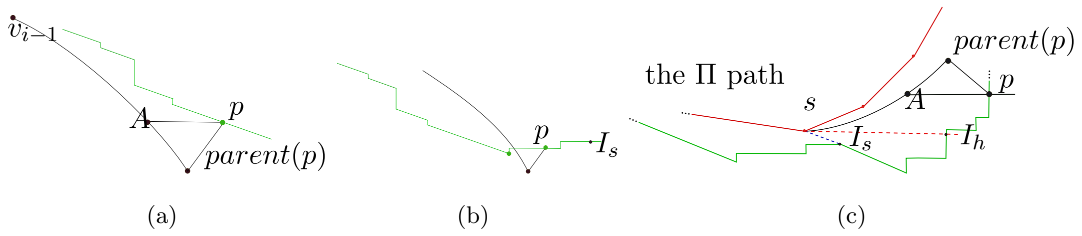


Figure 4.15: Proof of Lemma 4.8.3. The path Π is shown in dots.

4.8.2 Zig-Zag Guard

Lemma 4.8.4. *At the beginning of* Vertical Guard *invoked from* Zig-Zag Guard*, we have* $\sin\alpha \le h = y(c_{\text{aux}}) - y(p)$.

Proof. Recall that the pursuer performs vertical guard while it is in the 1st or the 3rd type critical sub-polygons (see Section 4.6). In the following, we present the proof for the 1st type.

Consider the preceding *Search*. Recall that during the search state p moves along the search path from v. The pursuer performs the zig-zag guard when e appears in its fourth quadrant (see Section 4.6). Recall that $c_{\text{aux}} = v$ (Section 4.6.3).

Suppose that $v \in Chain_U$. First, suppose that p is on the step-path. Then observe that e cannot force p to retreat beyond v because the step that p lies on is after the floor point and moreover the evader is confined in the corresponding step. Hence only by following the zig-zag moves the pursuer will catch up to $\pi(O_L, e)$.

Next suppose that p is on the α-path. Let l_1 be the distance p has traveled from v. Thus $l_1 \sin\alpha$ is the minimum height p obtains during search. Moreover $1 - l_1$ is the residual move which p travels downward during the *Zig Zag Guard*. Thus the total height p obtains during search phase is at least $h_{min} = l_1 \sin\alpha + 1 - l_1 = l_1(\sin\alpha - 1) + 1$ which is at least $\sin\alpha$. Recall that vertical guard is invoked in the case that initially p is above $\pi(O_L, e)$ and the zig-zag strategy is to move downward or to the left. In other words $y(c_{\text{aux}}) - y(p)$ increases afterward and hence the lower bound $\sin\alpha$ remains valid.

The above argument is valid when $v \in Chain_L$. □

Lemma 4.8.5. *Suppose that the pursuer starts Search state (on pocket$(v, \vec{r})$) in the* 1st *or the* 3rd *type critical sub-polygons but it finds e while it is inside the* 2nd *or the* 4th *type critical sub-polygons. Recall that here independent of the quadrant that e is inside, the pursuer invokes the zig-zag guard (Section 4.6). Then the evader* cannot *force p to retreat beyond v. In other words, only by following the zig-zag moves, p will start the next state S or L.*

Proof. First, note that the portion of the polygon formed by two consecutive α lines or horizontal lines, ∂Q and the vertical lines is a triangle, see Section 4.5.

Note that we omitted presenting the zig-zag moves for the 2nd or the 4th types. Let us start by presenting the detailed description of the zig-zag moves that p takes toward $\pi(O_L, e)$. According to Lemma 4.8.3, the slope of $parent(p)p$ is positive ($0 \leq slope$). Also, at the moment that e becomes visible $x(p) < x(e)$ (Observation 4.5.1). Recall that the search path in these types is composed of only the step-path (section 4.5). We present the strategy by dividing each step into two parts. Also we only present the argument for the 2nd type. See Fig. 4.16 and consider the step from A to D:

1. The segment uD: As a corollary of Lemma 4.8.3, all points on this segment are direct children of u. Therefore, we would have $parent(p) = u$.

   (a) If e is inside the fourth quadrant of p: See Fig. 4.16(a). Observe that the pocket formed by the segment uD and ∂Q from u to D is a simple pocket. By performing the simple pocket strategy presented in section 4.8.5, the evader is forced to cross uD while p is also on this segment. Note that uD is an edge of the shortest path tree. Note that all points on this segment are direct children of u. In other words, the next state will be L while p is in $h(v)$.

(b) If e is inside the first quadrant of p: See Fig. 4.16(b). Note that $\pi(O_L, e)$ has to be above p. Then p moves along $-\vec{X}_p$ toward $\pi(O_L, e)$. The L state or the S state will be established while p is in $hf(v)$.

2. From A to u: Note that e has to be in the first quadrant of p because otherwise p must have seen him sooner. The configuration that $\pi(O_L, e)$ is above p, shown in Fig. 4.16(b), is similar to the case (b) above. The configuration that $\pi(O_L, e)$ is below p, shown in Fig. 4.16(c), is as follows. The pursuer moves toward $\pi(O_L, e)$ along $\vec{X}_p$. This ensures that e is in the first quadrant of p until e crosses $\vec{X}_p$. At this time, p moves toward $\pi(O_L, e)$ along $-\vec{Y}_p$, see Fig. 4.16(d). Note that p is becoming closer and closer to $\pi(O_L, e)$ while $\pi(O_L, e)$ is confined in the triangular region $\triangle ABu$. Hence the next state (L or S) will be established while p is in $h(v)$.

□

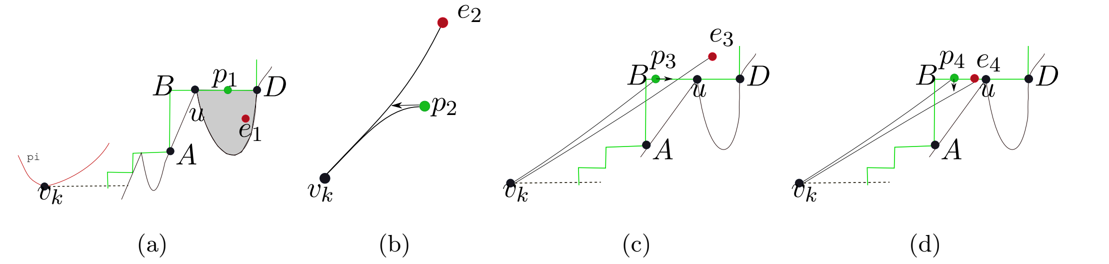


Figure 4.16: The zig-zag moves when p invokes zig-zag guard inside the 2^{nd} type critical sub-polygons while v in the preceding S state was in the 1^{st} type. See Lemma 4.8.5.

4.8.3 Simple Guard

In simple guard strategy, we define a local variable called the auxiliary vertex p_{aux} which is used as a landmark to guarantee progress. In simple guard state, the pursuer's goal is to prevent the evader from contaminating the region to the left of p_{aux}. In other words, the pursuer guarantees that the evader is inside $h(p_{\text{aux}})$. We define p_{aux} such that it is inside $h[p_{\text{ref}}]$. Therefore, at the end of this state the evader is inside $h(p_{\text{ref}})$.

Next let us present the selection of the vertex p_{aux}. Let *pocket*$(v, \vec{r})$ be the pocket which has been searched in the previous S state. Suppose that v_{aux} is the vertex that the α-path starts from (if $v \in Chain_U$, then $v_{\text{aux}} = v$). Moreover, let p_{ceil} be the ceiling point (refer to Section 4.5). Then:

- if p_0 is in the portion of the search path, from v to v_{aux}: See Fig. 4.17(a). Here p_{aux} is the bottommost vertex from the upper chain which is in the region in $h[p_{\text{ref}}]$ and to the left of the segment $p_0 p_{\text{ref}}$. Note that only when $v \in Chain_L$, we have $v_{\text{aux}} \neq v$ e.g. in Fig. 4.7(d) we have $v_{\text{aux}} = e_3$.

- If p_0 is in the portion of the search path from v_{aux} to the floor point: then p_{aux} is the first endpoint of the α-step that p_0 lies on it. For example, in Fig. 4.7(c), if p_0 is on the α-step defined from e_2 to I_2 then $p_{\text{aux}} = e_2$.

- If p_0 is in the portion of the search path after the floor point: See Fig. 4.17 parts (b) and (c). Let a be the intersection point between the α line passing through p_0 and Π. Suppose that $w_1 w_2$ and $w'_1 w'_2$ are the edges on Π such that $x(w_1) \leq x(p_{\text{ceil}}) < x(w_2)$ and $x(w'_1) \leq x(a) < x(w'_2)$ respectively. Then, if a is inside the next critical sub-polygon, p_{aux} is the second critical endpoint that defines the current critical sub-polygon. If $w_1 \neq w'_1$, i.e. a and p_{ceil} are not in between the endpoints of the same edge on Π, then $p_{\text{aux}} = w'_1$, see Fig. 4.17(b). Otherwise, p_{aux} is the bottommost vertex from the upper chain which is inside $h[p_{\text{ceil}}]$ and to the left of the line that connects p_{ceil} to p_0, see Fig. 4.17(c).

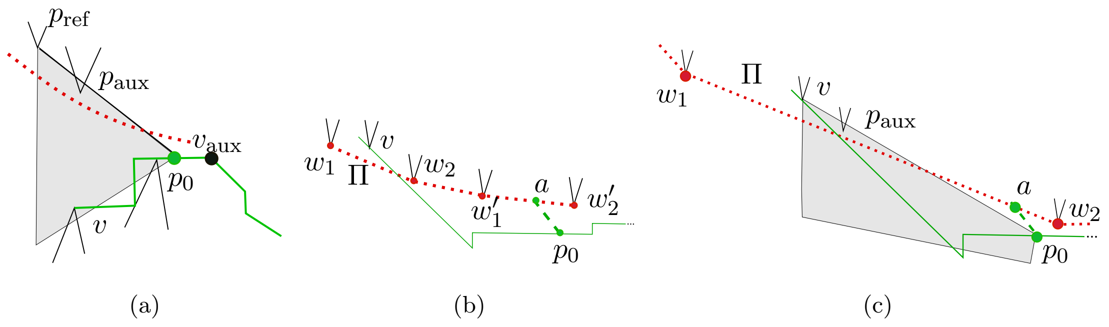


Figure 4.17: The path Π is shown in dots. Note that all upper chain vertices are above Π. (a) here p_{aux} is the bottommost vertex from $Chain_U$ in the shaded region. (b) here $p_{\text{aux}} = w'_1$ ($p_{\text{ceil}} = v$). (c) p_{aux} is the bottommost vertex from $Chain_U$ in the shaded region ($p_{\text{ceil}} = v$).

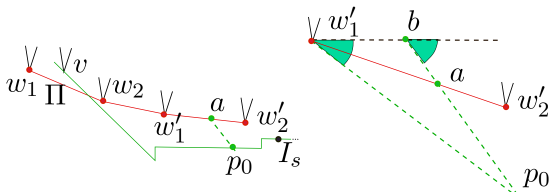


Figure 4.18: The point p_{aux} when $w_1 \neq w'_1$. Here $p_{\text{aux}} = w'_1$.

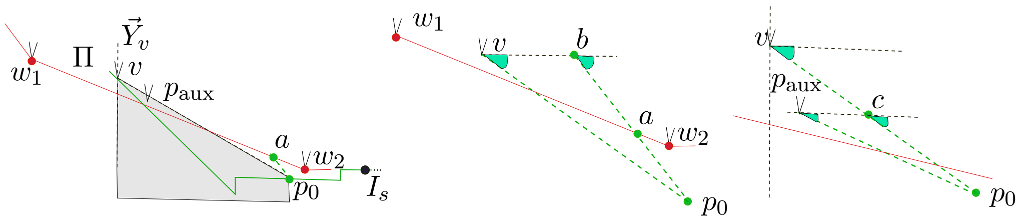


Figure 4.19: The point p_{aux} when $w_1 = w'_1$.

Lemma 4.8.6. *Suppose that the pursuer is in simple guard state and p_0 be the location of the pursuer at the beginning of the state. Then p_{aux} defined in section 4.6.2 is visible to p_0 and moreover, the angle built by p_0p_{aux} and the x-axis is less than α.*

Proof. Let $pocket(v, \vec{r})$ be the pocket being searched in the previous search state.

1. $v \in Chain_U$: here p_{aux} are defined based on the second and the third definition in section 4.6.2.

   - If p_0 is in the portion of the search path from v_{aux} to the floor point: since both p_0 and p_{aux} are on the α line of the corresponding α-step, the slope of p_0p_{aux} is equal to $-\alpha$.
   - If p_0 is in the portion of the search path after the floor point: first suppose that a and p_{ceil} are not in between the endpoints of the same edge on Π. Then $p_{\text{aux}} = w'_1$. See Fig. 4.18-right. Note that all upper chain vertices are above Π and all lower chain vertices are below the search path. Hence p_{aux} is visible to p_0. Next, let b be the intersection between p_0a and the horizontal line passing through w'_1. Note that the angle between this horizontal line and p_0b is equal to α. Also, observe that w'_1 is the left of p_0b. Hence, considering

the triangle $\triangle p_0 b w'_1$, it can be concluded that the angle between $p_0 p_{\text{aux}}$ and the x-axis must be less than α.

Next, suppose that a and p_{ceil} are in between the endpoints of the same edge on Π. Observe that p_{aux} is visible to p_0 as follows. This is because p_{aux} is the bottommost upper chain vertex in the shaded region, and moreover the slope of the edge $w'_1 w'_2$ is negative (Lemma 4.8.2), and all upper chain vertices are above Π and all lower chain vertices are below the search path. See Fig. 4.19. Next consider the angle between $p_0 p_{\text{ceil}}$ and the x-axis. Considering the triangle $\triangle p_0 p_{\text{ceil}} b$ and the fact that the angle between $p_0 a$ and the x-axis is equal to α and p_{ceil} is to the left of $p_0 a$, we can conclude that the angle between $p_0 p_{\text{ceil}}$ and the x-axis is smaller than α, see Fig. 4.19-middle. Now observe that p_{aux} is in the triangular region formed by $p_0 p_{\text{ceil}}$, the edge $w'_1 w'_2$, and $\vec{Y}_{p_{\text{ceil}}}$. Let c be the intersection between $p_0 p_{\text{ceil}}$ and the horizontal line passing through p_{aux}, see Fig. 4.19-right. Since the angle between this horizontal line and $p_0 c$ is smaller than α and considering the triangle $\triangle p_0 p_{\text{aux}} c$, we conclude that the angle between $p_0 p_{\text{aux}}$ and x-axis is smaller than α.

2. $v \in Chain_L$:

   (a) The slope of $\vec{r}$ is negative: recall that when e appears inside the first quadrant of p, the pursuer performs the simple guard (section 4.6.2). Also, recall that when $v \in Chain_L$ the search path starts by the step-path (section 4.5). Since the slope of $\vec{r}$ is negative, the evader can appear in the first quadrant of p only when p_0 is after v_{aux}. Similar to the above case, it can be shown that the angle between $p_0 p_{\text{aux}}$ and the x-axis is smaller than α.

   (b) The slope of $\vec{r}$ is positive: let us refer to the current simple guard state as G_2 and its corresponding search state which is on $pocket(v, \vec{r})$ as S_2. Let us refer to the state before S_2 as $state_{prev}$. Then $state_{prev}$ must be a simple guard. This is because during zig-zag guard, the evader remains inside the fourth quadrant of p and hence the resulting pocket ($pocket(v, \vec{r})$) would have negative slope. Also, if the previous state was L the resulting pocket would have negative slope (a corollary of Lemma 4.8.1). Let $state_{prev} = G_1$ which is a simple guard. Also let p'_{aux} be the auxiliary point defined in G_1, and p'

be the location of p at the beginning of G_1. Therefore, the sequence of states is $G_1S_2G_2$, the pursuer is moving toward p'_{aux} along $p'p'_{\text{aux}}$ during G_1, and v is to the right of the line $p'p'_{\text{aux}}$. Recall that at the beginning of S_2, we set $p_{\text{ref}} = p'_{\text{aux}}$ (Lemma 4.6.2).

Now consider p_{aux} defined in G_2 (section 4.6.2). If the second or the third definition applies, the proof is similar to the above cases. Hence suppose that p_{aux} is defined according to the first definition (i.e. p_0 is in between v and v_{aux}). We continue by an inductive argument as follows. Suppose that the angle between $p'p'_{\text{aux}}$ and the x-axis (the absolute value) is equal to or less than α. Since $p_{\text{ref}} = p'_{\text{aux}}$, and v is to the right of $p'p'_{\text{aux}}$ the slope of vp_{ref} is less than α. Recall that the search state in between v and v_{aux} is increasing in the x-coordinate and decreasing in the y coordinate, see section 4.5. Therefore, the slope of p_0p_{ref} is also less than α. See Fig. 4.20(a). Since p_{aux} is defined as the bottommost vertex in $h[p_{\text{ref}}]$ which is also to the left of p_0p_{ref}, it can be shown that the slope of p_0p_{aux} is less than α (Fig.4.19-right).

It remains to show that p_{aux} is visible to p_0. According to our inductive argument, $p_{\text{ref}} = p'_{\text{aux}}$ is visible to p'. For the sake of contradiction, let us assume that p_{aux} is not visible to p_0 and hence is blocked by a vertex namely v_b. We must have $v_b \in Chain_L$. Also it must that $x(p') < x(v_b) < x(v)$. See Fig. 4.20(b). Let p_g be the position of p during G_1 at which the evader the evader disappears behind v. Note that p_g is on the segment $p'p'_{\text{aux}}$ and moreover v is visible to p_g. Since all lower chain vertices are below the path formed by $p'_{\text{aux}}pg$, p_gv and the search path between v and p_0, the vertex v_b cannot block p_0p_{aux}. Contradiction.

$\square$

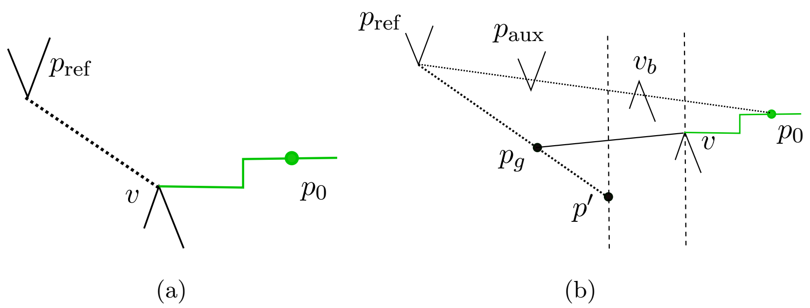


Figure 4.20: Proof of Lemma 4.8.6. (a) the angle $p_0 p_{\text{aux}}$ is less than α. (b) the vertex v_b has to be below the line $p_g v$.

Lemma 4.8.7. *Suppose that* p *is in the simple guard sub-state, see section 4.6.2. While the pursuer is moving back to* p_{aux} *the evader cannot cross the segment* $p_0 p_{\text{aux}}$.

Proof. Refer to Fig. 4.21-(a) let p_0 and e_0 be the position of the players at the beginning of the *Simple Guard*. Hence $x(p_0) < x(e_0)$ Observation 4.5.1. We will show that for all points A on the line segment between p_{aux} to p_0, the length of $p_0 A$ is smaller than the length of the shortest path from e_0 to A minus *one*. Hence if e tries to cross the $p_0 p_{\text{aux}}$ at A is will be captured by p. Specifically we show that $Ap_0 - Ae_0 \leq 1$. First observe that the angle $p_{\text{aux}} p_0$ is equal to or smaller than α, see Lemma 4.8.6. Let H be the point on Y_{p_0} where AH is perpendicular to Y_{p_0}.

1. $\alpha = \psi_1$: In this case we have $\cos\alpha = (1 - 1/D^2)^{0.5}, \quad Ap_0 \leq D, \quad 1 \leq D$. As a result, it must be that $Ap_0(1-\cos\alpha) \leq D(1-(1-1/D^2)^{0.5}) = D-(D^2-1)^{0.5} \leq 1$.

2. $\alpha = \psi_2 < \psi_1$: Here we have $\cos\psi_1 < \cos\psi_2, \quad -\cos\psi_2 = -\cos\alpha < -\cos\psi_1$. Therefore, $Ap_0(1-\cos\alpha) < Ap_0(1-\cos\psi_1) \leq 1$.

□

Lemma 4.8.8. *At the beginning of* Vertical Guard *invoked from* Simple Guard *we would have* $\sin\alpha \leq h = y(c_{\_aux}) - y(p)$.

Proof. Recall that here $c_{p_{\text{aux}}} = p_{\text{aux}}$. In the following, the key observation is that the angle between $p_{\text{aux}} p$ and the x-axis is smaller than α, see Lemma 4.8.6.

Note that e has to cross the line $p_{\text{aux}} p$ before crossing l_v. From $p_{\text{aux}} p$ to l_v the pursuer follows e by lion's move with respect to p_{aux}. Now consider the time-step that

p and e are on $p_{\text{aux}}p$. Let p' and e' be the position of the pursuer and the evader after one step of the lion's move with respect to p_{aux}. According to Lemma 4.8.9 the distance from all points on $p_{\text{aux}}p$ to all points on l_v is equal or greater than one. Therefore e' is a point in between $p_{\text{aux}}p$ and l_v see 4.21-(b).

1. $1 \leq p_{\text{aux}}p$: Because of the lion's progress $p_{\text{aux}}p < p_{\text{aux}}p'$. Hence $1 \leq p_{\text{aux}}p'$. Moreover p' is in between the $p_{\text{aux}}p$ and l_v. Thus $\sin\alpha \leq h = y(p_{\text{aux}}) - y(p')$ see Fig. 4.21-(c).

2. $p_{\text{aux}}p \leq 1$: Let $A = \vec{X}_p \cap \vec{Y}_{p'}$. There are three cases:

   (a) p' is outside the unit circle: Here, similar to the first case we have $\sin\alpha \leq h$.

   (b) $x(p') < x(p)$, see Fig. 4.21-(e) and (f) for p_2': We have $h = Ap_2'$. Thus, $\frac{Ap}{p_{\text{aux}}p} < \cos\alpha$. Hence, $p_{\text{aux}}p < 1, Ap \leq p_{\text{aux}}p \cdot \cos\alpha < \cos\alpha$. Therefore, $Ap < \cos\alpha, \qquad h^2 = 1 - Ap^2$. Thus, we have $\sin\alpha < h$.

   (c) $x(p) < x(p')$, see Fig. 4.21-(d) and (g) for p_1': In this case, we have $\beta \leq \frac{\pi}{2} - \alpha, \sin\alpha \leq \cos\beta, pp_1' = 1, h = \cos\beta$. Thus, $\sin\alpha < h$.

□

Lemma 4.8.9. *Let* p_{aux} *be a point inside* Q*. Consider the* α*-line passing through* p_{aux}*, the unit circle centered at* p_{aux}*, and the vertical line* $l_v = \vec{Y}_{p_{\text{aux}}}$*. For all points* e *above the* α*-line and all points* e' *on* l_v *in which* e *and* e' *are outside the unit circle we would have* $1 \leq ee'$*.*

Proof. Refer to Fig. 4.21-(b) note that $ee' = \tan\theta_1 + \tan\theta_2$ where $\theta_1 + \theta_2 = \pi - \alpha$. The function $\tan\theta_1 + \tan\theta_2 - 1$ is positive for α angles equal or smaller than $\psi_2 = (\frac{\pi}{2} - 2\arctan\frac{1}{2})$. Hence $1 \leq ee'$ for α which is equal or smaller than ψ_2. See Definition 4.5.2. □

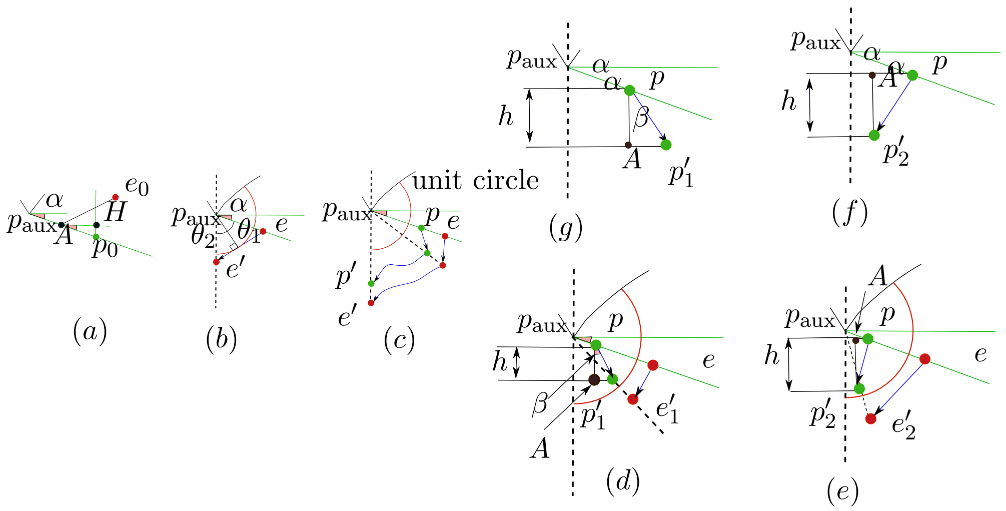


Figure 4.21: e cannot cross $p_{\text{aux}}p$ during simple guard

4.8.4 Vertical Guard and Horizontal Guard

Lemma 4.8.10. *The vertical guard circle centered at c prevents e from escaping to upper chain vertices which are to the left of p_{ref}.*

Proof. See Lemma 4.8.13. □

Lemma 4.8.11. *If during vertical guard strategy e disappears behind a vertex to the left of l_v, then the resulting pocket would be a simple pocket. Also refer to Fig. 4.12 parts (b) and (c).*

Proof. The vertex that defines the pocket must be from the lower chain (Lemma 4.8.10). In this figure, because of monotonicity the pocket in the left is impossible and hence the pocket must be an simple pocket (middle part of the figure). □

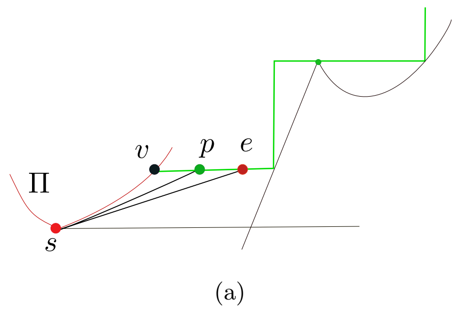
Lemma 4.8.12. Feasibility of the Horizontal Guard. *Suppose that p invokes the horizontal guard sub-state. Then, the radius of the circle centered at c is finite (i.e. upper bounded) and the pursuer can perform lion's move with respect to c.*

Proof. Recall that the pursuer performs the horizontal guard strategy when: (1) the previous search state was on $pocket(v, \vec{r})$ where v is inside the 2^{nd} or the 4^{th} type critical sub-polygon, and (2) $x(v) \leq x(p) < x(e)$ and $y(p) = y(e) = y(v)$ (Lemma 4.6.1).

Note that $x(v) \leq x(p) < x(e)$ and $y(p) = y(e) = y(v)$ can occur only when the evader appears while p is on the horizontal line passing through v (i.e. the first step) (see Fig. 4.22). If p is on other steps, similar to Lemma 4.8.5 we can show that the zig-zag moves are sufficient.

Also note that since the slope of the entrance $\vec{r}$ is positive, at the beginning of G state (end of the S state) the evader would be in the first quadrant of p.

Similar to Lemma 4.6.3, it can shown that the radius of the circle centered at c is upper bound if we could provide a lower bound for $x(p)-x(v)$. Note that $1 \leq x(p)-x(v)$. This is because during search p is moving in the direction of the x-axis (the horizontal line passing through v which is on the first step). Even if the evader appears as p moves for $\epsilon < 1$ (during the one time unit of the S state), the pursuer immediately switches to the G state and moves for the residual move toward $\pi(O_L, e)$ (the residual move is $(1-\epsilon)$). Therefore at the time that e crosses $\vec{X}_v$, we would have $1 \leq x(p) - x(v)$. $\square$



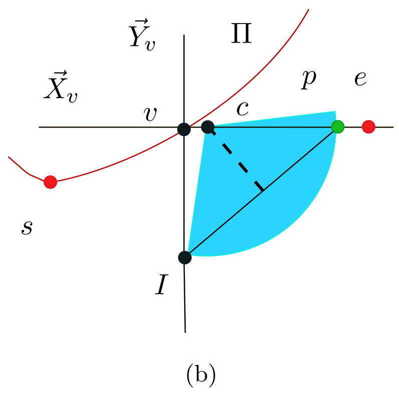


Figure 4.22: (a) The configuration that p performs the horizontal guard strategy. (b) The horizontal guard circle centered at c.

Lemma 4.8.13. Feasibility of the Vertical Guard. *Consider the vertical guard state, section 4.6.3. Then all upper chain vertices before* c_{aux} *are above* $c_{\text{aux}}I$. *Therefore,* p *can perform the lion's move with respect to* c_{aux} *and* c. *In other words, the next location that* p *must move to according to the lion's move is in the free space.*

Proof. Note that the vertical guard state can be invoked from zig-zag guard, simple guard. Therefore, $c_{\text{aux}} = v$ (in case of zig-zag guard) or $c_{\text{aux}} = p_{\text{aux}}$ (in case of simple guard).

In the following, we present the proof by arguing the 1^{st} and the 2^{nd} type critical sub-polygons. The other two types are symmetric.

1. $c_{\text{aux}} = v$ (i.e. the vertical guard is invoked from zig-zag guard): Here, v must be a vertex inside the 1^{st} type and moreover at the beginning of the G state p must be inside the same critical sub-polygon (see section 4.6 and Lemma 4.8.5). Also note that $v \in Chain_U$. To see this suppose that $v \in Chain_L$. Recall that the search path on $pocket(v, \vec{r})$ starts by the step-path, then continues along the α-path and then the step-path (section 4.5). Let I_1 be the intersection of the first step-path with ∂Q. Recall that p performs the zig-zag guard when e appears inside its fourth quadrant (section 4.6.1). Suppose that p is in between v and I_1. According to zig-zag guard, the pursuer moves downward and to the left. Hence the evader cannot force the pursuer to retreat beyond v (the players will hit ∂Q). When p is after I_1 the similar result is valid.

   Therefore only when $v \in Chain_U$, the pursuer invokes the vertical guard during zig-zag guard. In the following we prove that all upper chain vertices before v are above vI. Let us refer to the current G state as G_2 and the S state before it (which is on $pocket(v, \vec{r})$) as S_2. Also, let $state_{prev}$ be the state before S_2. Hence the sequence of states is $state_{prev}S_2G_2$.

   (a) $state_{prev} = L$: Let w_1w_2 be the edge on Π so that $x(w_1) \leq x(v) < x(w_2)$. Then v must be in the fourth quadrant of w_1. Because otherwise $pocket(v, \vec{r})$ would be a simple pocket and p can recover the L state by following the simple pocket strategy presented in section 4.8.5. Since the slope of edges on Π before w_1 is negative (Lemma 4.8.2) it must be that all upper chain vertices before v are above vI.

   (b) $state_{prev} = G$: Let us denote this G state as G_1 and its previous S state as S_1. Hence the sequence of states is $S_1G_1S_2G_2$. Also suppose that S_1 is on $pocket(v_1, \vec{r_1})$.

      i. G_1 is zig-zag guard and $v_1 \in Chain_U$: We continue by an inductive argument as follows. Suppose that for all invokes to the S state before

S_2 all upper chain vertices are above the corresponding horizontal line. In other words, all upper chain vertices before v_1 (in S_1) which are before v_1 are above the horizontal line passing through v_1. Note that during S_1, the pursuer moves downward and to the right. During G_1, which is a zig-zag guard, the pursuer moves downward and to the left and moreover the evader remains inside the fourth quadrant of the pursuer. See Fig. 4.23(a). Hence all upper chain vertices are above the horizontal line passing through v.

Now we prove the property for the first invoke to G. Therefore, the sequence of states is $S_2G_2...$ where S_2 is the first search state which is done on $pocket(v, \vec{r})$. Observe that the first time that the pursuer invokes S in a critical sub-polygon, the vertex $v \in \Pi$. Even if the sequence of states is $LS_2G_2...$, either $pocket(v, \vec{r})$ is a simple pocket which p can recover the L state by following the simple pocket strategy, or the property must hold for v.

ii. G_1 is zig-zag guard and $v_1 \in Chain_L$: Likewise, since all upper chain vertices are above the search path and the blocking vertex v is inside the fourth quadrant of p's location during G_1. See Fig. 4.23(b).

iii. G_1 is simple guard: Recall that $v \in Chain_U$. According to the simple guard the entrance of $pocket(v, \vec{r})$ is $\vec{r} = p_{\text{aux}}v$ where p_{aux} is the auxiliary reference point defined in G_1. Note that v must be inside the fourth quadrant of p_{aux} because otherwise $pocket(v, \vec{r})$ would be a simple pocket and by following the simple pocket strategy (section 4.8.5) the pursuer will recover the simple guard strategy.

In the following we show that in simple guard all upper chain vertices before p_{aux} are above the horizontal line passing through p_{aux}. Therefore all upper chain vertices before v are above the horizontal line passing through v. See Fig. 4.23(c).

(c) $c_{\text{aux}} = p_{\text{aux}}$ (i.e. the vertical guard is invoked from simple guard): By an inductive argument we show that all upper chain vertices before p_{ref} are above the horizontal line passing through p_{ref}. Using this we show that all upper chain vertices before p_{aux} are above its corresponding horizontal line.

In our inductive argument we also use the result obtained above: all upper chain vertices are above v if $v \in Chain_U$ and v is defining $pocket(v, \vec{r})$ being searched in an S state.

Suppose that the property holds for the current p_{ref}. Then referring to section 4.6.2, the point p_{aux} is defined based on the location of p at the beginning of the simple guard. Let us denote the current guard state as G_2 and the previous search state as S_2 which is performed on $pocket(v, \vec{r})$. Also recall that p_0 is the location of p at the beginning of G_2 (section 4.6.2).

- If p_0 is in the portion of the search path, from v to v_{aux}: Recall that here $v \in Chain_L$ and p_{aux} is the bottommost vertex from the upper chain which is in the region in $h[p_{\text{ref}}]$ and to the left of the ray p_0p_{ref}. See Fig. 4.24(a). Since p_{aux} is the bottommost vertex in this region and all upper chain vertices are above the horizontal line passing through p_{ref}, the property also holds for p_{aux}. Also recall that we update p_{ref} to p_{aux} at the end of the simple guard (Lemma 4.6.2). Therefore, the property is valid for the next p_{ref} point.
- If p_0 is in the portion of the search path from v_{aux} to the floor point: First, suppose that $v \in Chain_L$. Since all upper chain vertices are above the search path, the property holds for p_{aux}. Next suppose that $v \in Chain_U$. Since the y coordinate of the points on the search path is decreasing and all upper chain vertices are above the search path and the property holds for v, we conclude that the property also holds for p_{aux}.
- If p_0 is in the portion of the search path after the floor point: Recall that if $w_1 \neq w'_1$, i.e. a and p_{ceil} are not in between the endpoints of the same edge on Π, then $p_{\text{aux}} = w'_1$, see Fig. 4.24(b). Note that since $w'_1 \in \Pi$, the property holds for $p_{\text{aux}} = w'_1$.
  If $w_1 = w'_1$, i.e. a and p_{ceil} are in between the endpoints of the same edge on Π, then p_{aux} is the bottommost vertex from the upper chain which is inside $h[p_{\text{ceil}}]$ and to the left of the line connecting p_{ceil} to p_0, see Fig. 4.24(c). Here, v could be from the lower chain or the upper chain. In the case that $v \in Chain_L$, since all upper chain vertices are above the search path the property will be concluded. When $v \in Chain_U$, since the

property is valid for v and p_{aux} is the bottommost vertex, the property is concluded.

$\square$

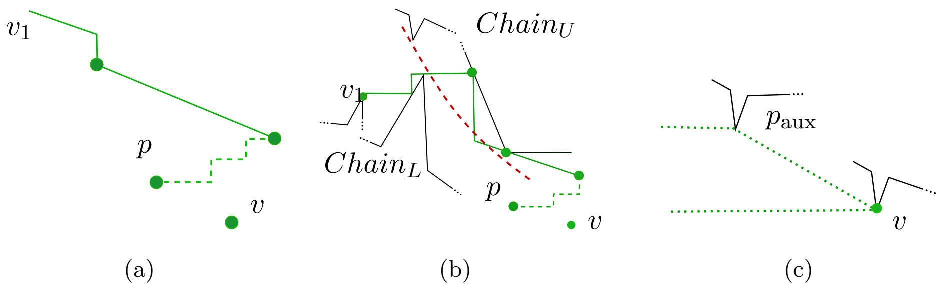


Figure 4.23: All upper chain vertices are above vI. The search path on v_1 is shown in green and the path traversed during G_1 is shown dash green lines. (a) $v_1 \in Chain_U$. Here G_1 is a zig-zag guard. (b) $v_1 \in Chain_L$. Here G_1 is a zig-zag guard. (c) Here G_1 is a simple guard.

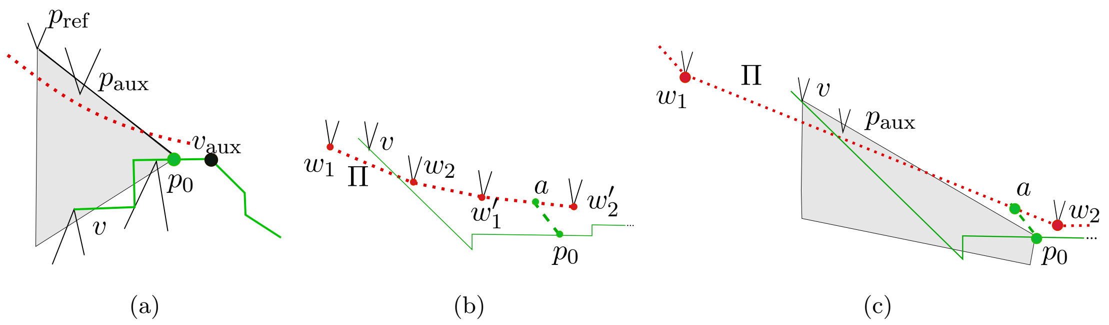


Figure 4.24: Fig. 4.17 shown for convenience. The path Π is shown in dots. Note that all upper chain vertices are above Π. (a) Here p_{aux} is the bottommost vertex from $Chain_U$ in the shaded region. (b) Here $p_{\text{aux}} = w'_1$ ($p_{\text{ceil}} = v$). (c) p_{aux} is the bottommost vertex from $Chain_U$ in the shaded region ($p_{\text{ceil}} = v$).

4.8.5 Simple Pockets

The pocket $pocket(v, \vec{r})$ is called a *simple pocket* if its boundary except the entrance $\vec{r}$, is a single x-monotone chain and the angle between $\vec{r}$ and the y axis is smaller than $\frac{\pi}{2}$.

The pursuer by following the MPC strategy described in this paper, can force the evader to exit the pocket in order to prevent capture. The only difference is that during search it is sufficient to move along $\vec{r}$ in order to find e. As the evader is found the pursuer starts the simple guard strategy presented in Sect. 4.6.2 i.e. it moves toward v along $\vec{r}$. Note that e cannot cross the segment between v and p because of the slope.

Lemma 4.8.14 (Simple Pockets)**.** *By following the MPC pursuit strategy on a simple pocket $pocket(v, \vec{r})$, after at most $O(n^2D^3)$ time-steps, e is forced to cross the entrance and exit the pocket in order to prevent being captured. At the crossing time, p and e both lie on $\vec{r}$ and p is in between v and e, see Fig. 4.12-(d). The only difference is that during search the pursuer moves along $\vec{r}$.*

Proof. First observe that p will eventually stop performing the current state, which can be one of: *Search, Guard* or *Lion*, and switch to the next state. Suppose that the current state is *Guard*. Then the next state can be either *Lion* or *Search*.

In the former case, the pursuer gains lion's progress with respect to v until e is captured or e disappears behind a vertex v' in the new pocket $pocket(v', \overrightarrow{pv'})$ or e exits the initial pocket.

In the latter case, the vertex v' which defines the new pocket $pocket(v', \overrightarrow{vv'})$ has the property that $x(v) < x(v')$. Hence the new pocket is a smaller one contained in the original pocket.

To complete the proof, note that once the evader hides behind a vertex, say v, it cannot disappear behind the vertex for the second time.

Since there are n vertices and the entrance is traversed twice, once during search and once during guard, the total time spent in the guard and search states would be $O(2nD)$. Together with $O(nD^2)$ required for extended lion's move progress, the total capture time would be $O((nD^2) \cdot (2nD)) = O(n^2D^3)$.

It remains to show that the entrance of all possible new pockets is positive and hence we can recursively use the simple pocket strategy. Clearly, all possible new pockets after the *Guard* state has positive slope since v' is inside $pocket(v, \vec{r})$.

Next, we show that all possible pockets after the lions move state must have positive slope. Since during lion's move, p lies on the edge $parent(e)e$, the entrance $\overrightarrow{pv'}$ is in direction of the tree edge $parent(v')v'$. See Fig. 4.25(a). Therefore, it is enough to show that in the shortest path tree rooted at v all edges have positive slope and hence if the evader disappears during *Lion's* state, the new pocket $pocket(v', \overrightarrow{pv'})$ will have positive slope entrance (the entrance is $\overrightarrow{pv'}$).

Suppose that there is an edge uw on the shortest path tree rooted at v with negative slope i.e. u is the parent of w in $\pi(v, w)$. We will show that there exists a path from v to w which is shorter than $\pi(v, w)$ which is a contradiction. See Fig. 4.25(c). Since the original pocket had a positive slope, there exists a vertex u' below vw such that w is visible to u'. Note that u' can be v itself. Also since w is visible to u, it must be that $x(u') < x(u)$ because otherwise u' would block the edge uw (note that slope of uw is negative and the slope of $u'w$ is positive). Then according to the triangulation property the shortest path from v to u' followed by the edge $u'w$ yields a shorter path than $\pi(v, w) = \pi(v, u) + uw$. A contradiction. □

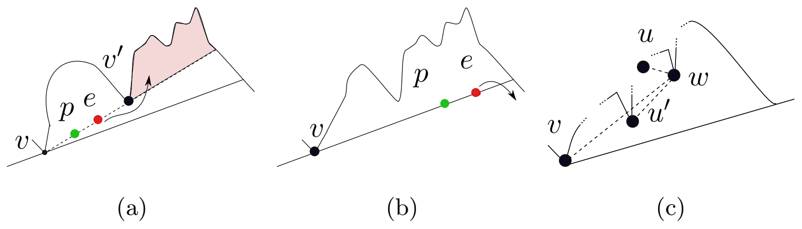


Figure 4.25: (a) The evader disappears into the shaded pocket. (b) Result of the simple pocket strategy. (c) Proof of Lemma 4.8.14.

4.9 Concluding Remarks

In this chapter, we showed that a single deterministic pursuer with line-of-sight visibility can capture an evader whose speed is equal to the pursuer's in any monotone polygon. We present our strategy on x-monotone polygons where the x coordinate of the points along the boundary increases from the left-most point to the right-most point. Our proposed strategy has three states: search, guard, and lion's move. In the search state, the evader is invisible and the goal of the search sub-strategy is to find the evader. In the lion's move state, the evader is visible and furthermore the pursuer is on the shortest path between the evader and the left-most point on the boundary of the polygon. As a result, the pursuer can perform the lion's strategy and make progress. The guard state is an intermediate state between search and lion's move, which enables the pursuer to locate itself on the shortest path to the evader. In other words, the guard state enables the pursuer to chase the evader by lion's move. As the pursuer is performing its strategy it is possible that it retreats back to the left. We show that even though the pursuer is retreating, it is making progress and hence the evader is captured in finite time.

In the next chapter, we study the full-visibility version of the pursuit-evasion game where the players are aware of each other's location at all times. We will study the game on three-dimensional surfaces.

Chapter 5

Pursuit-Evasion Games with Full-Visibility

In Chapter 4, we studied a variant of the pursuit-evasion game where the pursuer has line-of-sight vision in a monotone polygon. In this chapter, we turn our attention to the full-visibility case, i.e., when both players have complete knowledge of the location of their opponent. We study the game when played in a more complex environment: the surface of polyhedrons. First, in Section 5.2 we investigate the game on general polyhedral surfaces. We show that there exists a finite-time capture strategy with three pursuers even in the presence of obstacles. Then, in Section 5.3 we show that surfaces of convex terrains (height-maps) are single-pursuer-win.

Let us start this chapter by presenting the game model and the notation used throughout the chapter.

5.1 Game Model and Notation

A team of a finite number of pursuers, denoted by $p_1, p_2, ..., p_n$ wish to capture an evader which is denoted by e. When our pursuit strategy involves only one pursuer (in Section 5.3), we drop the subscript and denote the pursuer by p.

Throughout this chapter, we consider the discrete time turn-based version of the lion and man game: The players take turns and each turn takes a unit time step. At the beginning of the game, first the pursuers choose their initial positions and then the evader places itself. Afterwards, the pursuers move first, followed by the evader's move. The pursuers can move simultaneously or sequentially as long as they finish their moves within their turn. In one time step each player can move along any path contained in the environment which is at most one unit in length. The players can observe each other's locations at all times. The pursuers' goal is to capture the evader in finite time. The evader is captured if at any time the distance between some pursuer p_i and the evader is less than the **capture distance** δ. The pursuers win the game if they can capture the evader in finite time, while the evader wins if it can escape forever.

The game environment is denoted by $\mathcal{S}$. Roughly speaking, $\mathcal{S}$ is the surface of a polyhedron. Later in each section we will give a more accurate specification of the environment we study (polyhedral surfaces with obstacles in Section 5.2, and convex height-maps in Section 5.3). The surface $\mathcal{S}$ is represented by a set of faces f_i, a set of edges e_i, and a set of vertices v_i (Fig. 5.1(c)). We assume that each face f_i is triangulated, which can be done in time $O(n \log n)$ where n is the number of vertices on a polygonal face [78]. Thus, each face f_i is a triangle. An edge joins exactly two vertices of the polyhedron. A boundary edge lies on exactly one face, while a non-boundary edge is on two adjacent faces.

We denote the boundary of a set $R \subseteq \mathcal{S}$ by ∂R.

An arbitrary path on the surface, which is not necessarily a shortest path, is denoted by Π. For a given path Π on $\mathcal{S}$, we use $|\Pi|$ to denote the length of Π. A geodesic shortest path between points a and b is denoted by $\Pi^*(a, b)$. The length of $\Pi^*(a, b)$ is denoted by $d(a, b)$. We drop (a, b) if the source and destination points of Π^* are clear from the context. Given two points p and q on Π we denote the sub-path of Π from p to q by $\Pi(p, q)$. Given two paths $\Pi_1(a, q)$ and $\Pi_2(q, b)$, we denote the path obtained by concatenating them at q by $\Pi_1(a, q) + \Pi_2(q, b)$. Given two points p and q on the same face of $\mathcal{S}$, we denote the line segment between them by pq and its length by $|pq|$.

We are now ready to start our investigation of the game on polyhedral surfaces.

5.2 Three Pursuers on Polyhedral Surfaces with Obstacles

In this section, we show that three pursuers suffice for capture on a polyhedral surface (Fig. 5.1(a)) when the capture distance is non-zero. We show that our pursuit strategy is applicable even if the surface does not have boundary, and also in the presence of "obstacles" i.e. subsets of the surface that neither player can enter. As an example, Fig. 5.1(b) shows a geodesic terrain which is a special case characterized by unique height values for points in the two-dimensional plane.

To be more precise, in this section the environment $\mathcal{S}$ can be the surface of any polyhedron possibly with obstacles. Obstacles are forbidden subsets of $\mathcal{S}$ that neither the pursuers nor the evader can enter. Topologically, the surface in the absence of obstacles may be equivalent to a disk, in which case it has a boundary, or a sphere, in which case it does not have a boundary. Each obstacle is homeomorphic to a disk in $\mathbb{R}^2$. Obstacles are added by punching holes on the surface without introducing handles. We denote the set of obstacles on $\mathcal{S}$ by $\{O_1, O_2, ..., O_M\}$.

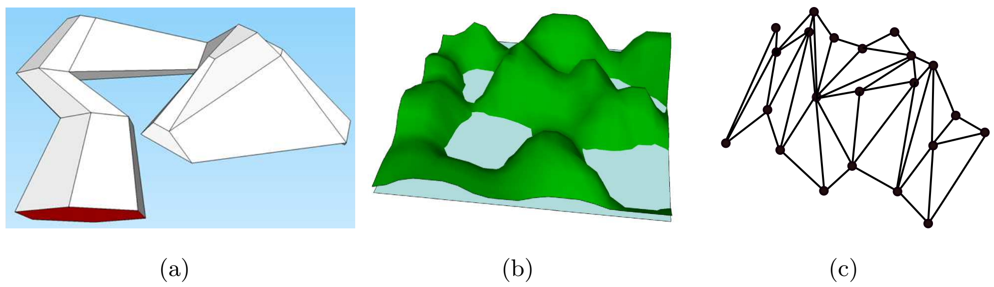


Figure 5.1: Illustration of the key concepts. (a) A polyhedral surface which is not a terrain. When the red face is excluded the surface has a boundary, and thus is homeomorphic to a disk. If it is included in $\mathcal{S}$, the surface is homeomorphic to the surface of a sphere. (b) A geodesic terrain. When the players are restricted to stay outside water, the lakes are modeled as obstacles. (c) A triangulated polyhedral surface. Here, faces are the triangles, edges are shown as line segments and vertices are depicted as dots.

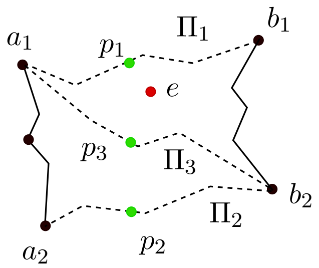
We present a pursuit strategy which uses a subroutine for guarding shortest paths inspired by Aigner and Fromme [28] (see Chapter 3). We show how the evader can be constrained to a region formed by two shortest paths each of which is guarded by a pursuer. We use the third pursuer to split this region to two smaller regions such that one of them contains the evader. The splitting algorithm is applied iteratively. We prove that the evader will be captured in finite time by obtaining a lower-bound on how much the evader's region shrinks at each iteration.

This section is organized as follows. In Section 5.2.1 we discuss the required modules for our strategy. The details of the pursuit strategy are given in Section 5.2.2. In Section 5.2.3 we show that the evader is captured in finite time. Finally, in Section 5.2.4 we present the lemmas required for the analysis of the correctness of our strategy.

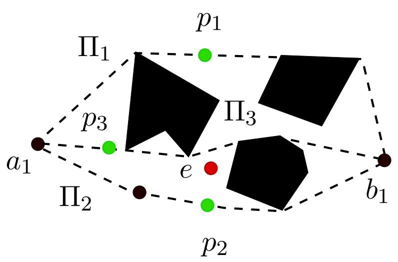
5.2.1 Ingredients of the Pursuit Strategy

At a high level, our three pursuer strategy for capturing the evader on $\mathcal{S}$ is similar to the strategy proposed in [38] for polygonal environments with obstacles. Let us start by presenting the idea for polygons. The pursuit strategy is divided into phases. In each phase, the pursuers make progress by restricting the evader to a smaller region. Let us refer to the subset of $\mathcal{S}$ that the evader is restricted to during the i^{th} phase as the **contaminated region** and denote it by $\mathcal{S}_i$. The strategy has two components: **guarding** and **splitting**. Two pursuers guard the boundary of the current contaminated region $\mathcal{S}_i$ in order to prevent the evader from exiting $\mathcal{S}_i$. A pursuer can guard a shortest path Π by locating itself on the **projection** of the evader onto Π. The evader cannot cross Π without being captured by the guarding pursuer. Therefore, we maintain the invariant that the subsets of $\partial\mathcal{S}_i$ that are guarded by two pursuers are shortest paths in $\mathcal{S}_i$.

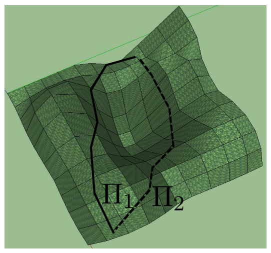
Since only two pursuers are used to guard $\mathcal{S}_i$, the third pursuer is free. This third pursuer splits $\mathcal{S}_i$ into two smaller regions by guarding a third shortest path inside $\mathcal{S}_i$. The splitting shortest path is selected in such a way that one of the two pursuers guarding $\mathcal{S}_i$ becomes free. This allows the pursuers to continue the splitting process until capture. The evader will be now restricted to one of the resulting smaller regions inside $\mathcal{S}_i$ which is denoted by $\mathcal{S}_{i+1}$. The evader can never return to $\mathcal{S}_i \setminus \mathcal{S}_{i+1}$. Furthermore, using the free pursuer the splitting process can be continued on the new contaminated region $\mathcal{S}_{i+1}$.



(a)



(b)



(c)

Figure 5.2: The paths Π_1, Π_2, and Π_3 are shown with dashed lines and $\partial\mathcal{S}$ is shown with solid lines. (a) The path Π_3 is selected as the shortest path between a_1 and b_2 when $a_1 \neq a_2, b_2$ and $b_1 \neq a_2, b_2$. (b) In polygons multiple shortest paths are possible only because of obstacles. (c) On a polyhedron multiple shortest paths can be built by hills and valleys.

The critical parts of the strategy are choosing the path for the splitting procedure and showing that progress toward capture is guaranteed after each splitting step. There are significant differences between the polygonal and polyhedral cases involving these steps. First, let us explain the difficulties in the splitting step. Suppose that $\mathcal{S}$ has a boundary. Let Π_1 and Π_2 be the two paths on $\partial\mathcal{S}_i$ that are guarded by pursuers p_1 and p_2 respectively. Let a_1 and b_1 be the two endpoints of Π_1, and a_2 and b_2 be the endpoints of Π_2. Initially, a_i, b_i are chosen on $\partial\mathcal{S}$. If these endpoints are disjoint, i.e $a_1 \neq a_2, b_2$ *and* $b_1 \neq a_2, b_2$, then it is not too difficult to see that a path $\Pi(a_1, b_2)$, which is a shortest path inside $\mathcal{S}_i$, can be used for splitting (Fig. 5.2(a)). Next, suppose $a_1 = a_2$ but $b_1 \neq b_2$. In this case, we can pick a point c along the portion of the boundary from b_1 to b_2 and use a path $\Pi(a_1, c)$, which is a shortest path inside $\mathcal{S}_i$, to make progress.

The remaining case $a_1 = a_2$ and $b_1 = b_2$, i.e. when there are two shortest paths between $a_1 = a_2$ and $b_1 = b_2$, is challenging on a polyhedron (Fig. 5.2(c)). In particular, after removing Π_1 and Π_2, the next-shortest path between a_1 and b_1 can be infinitesimally close to either Π_1 or Π_2. Let us refer this candidate path as Π_{min} (Fig. 5.3). If we choose Π_3 as Π_{min}, then the splitting procedure may go on forever and we cannot have finite time capture. On the other hand, if we choose any other path as Π_3, then p_3 may not guard Π_3. This is because the evader can cross Π_3 by traveling along Π_{min}

since Π_{min} is shorter than Π_3.

In polygonal settings, multiple shortest paths between two points are possible only when they touch obstacles (Fig. 5.2(b)). In this case, the contaminated region is disconnected and we can make progress by removing components that do not contain the evader.

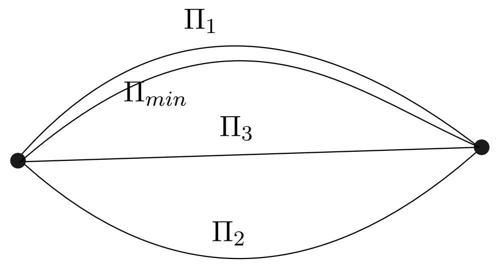


Figure 5.3: After removing Π_1, Π_2, the next shortest path Π_{min} can be very close to Π_1, Π_2.

In order to find the splitting path in the case of multiple shortest paths, we make use of the capture distance δ of the pursuers. In particular, for each of the shortest paths Π_i we define a **capture region** as a region around Π_i that the evader cannot enter without being captured by the guarding pursuer p_i. Let $C(\Pi_i)$ denote the capture region of Π_i. We use the intersection points between $\partial C(\Pi_1)$ and $\partial C(\Pi_2)$ as the endpoints of the splitting path (Section 5.2.2).

In order to show that the number of splitting phases is finite, the first idea is to provide a lower bound on $(\text{area}(\mathcal{S}_i) - \text{area}(\mathcal{S}_{i+1}))$. In particular, it must be shown that there exists a constant number $\epsilon > 0$ such that $(\text{area}(\mathcal{S}_i) - \text{area}(\mathcal{S}_{i+1})) \geq \epsilon$. Instead of directly computing the area, which is hard, we partition the surface of the polyhedron into $\frac{\delta}{2}$**-small triangles**, that is, triangles with all sides shorter than $\frac{\delta}{2}$. We show that at the end of each phase the pursuers claim at least one of these triangles as **cleared** for the rest of the game, i.e. the evader cannot return to the cleared triangles. This provides a lower bound on the progress.

Our result is the following theorem which we will prove in Section 5.2.3.

Theorem 5.2.1 (Finite time capture with three pursuers)**.** *In the lion and man game on a polyhedral surface $\mathcal{S}$, possibly with holes, three pursuers with non-zero capture* *distance δ can capture the evader in time $O((\frac{A}{\delta^2}+\frac{L}{\delta}+N)^2\frac{\delta}{2})$ where A denotes the area of the surface, L is the sum of the lengths of the edges on $\mathcal{S}$, and N is the number of triangular faces used in the representation of $\mathcal{S}$.*

In the following, we formally present the definitions of *projections* used for guarding the evader region, *capture regions* used for shrinking the evader region, and *small triangles* used to prove capture.

5.2.1.1 Projection

In Section 3.1.7 we introduced the concept of projection mapping. Recall that the projection of the evader onto Π^* is a point on Π^* which is closer to each point on Π^* than the evader. Aigner and Fromme [28] discussed guarding of shortest paths on graphs using projections. This idea was later used for guarding shortest paths in polygonal [38] and polyhedral environments [43]. Inspired from the guarding strategies in [28] and [38], Klein and Suri [43] show that shortest paths on polyhedral surfaces are guardable using the following projection mapping:

Definition 3 (Projection mapping [38]). *Given a shortest path $\Pi^*(a,b)$ between two points a and b, and the current evader location e, a point $p(e)$ on $\Pi^*(a,b)$ is a projection mapping of e onto Π^* if $d(a,p(e)) = \min(d(a,e),d(a,b))$.*

In order to guard $\Pi^*(a,b)$, the pursuer can locate itself on the projection of the evader as follows. The pursuer starts from a and moves along $\Pi^*(a,b)$ towards b. As the evader moves its projection gets closer or farther away from the pursuer. Since the pursuer is moving in the same direction, it will eventually reach the evader's projection. The number of required steps is bounded by $O(D+d(a,b))$ where D is the length of the longest shortest path on $\mathcal{S}$. Here, the $O(D)$ component accounts for the number of steps required for the pursuer to initialize itself on a, and the $d(a,b)$ component accounts for the number of steps required to travel along the path $\Pi^*(a,b)$ from a to b. The total number of steps is $O(D)$ because $d(a,b) \leq D$.

When the pursuer establishes its location on the evader's projection, it can maintain its position on the projection of the evader as the evader moves. In other words, as the evader moves from e_t to e_{t+1}, the guarding pursuer can move from the projection of e_t to the projection of e_{t+1} in one step. This is because $d(a,p(e_t)) = \min(d(a,e_t),d(a,b))$,

$d(a, p(e_{t+1})) = \min(d(a, e_{t+1}), d(a, b))$, and moreover $d(a, e_{t+1}) \leq d(a, e_t) + d(e_t, e_{t+1}) \leq d(a, e_t) + 1$. For a more elaborate proof, see [38].

The second important property of the projection is that since the projection is closer to all the points on Π^* than the evader itself, the evader cannot cross Π^* without being captured [38].

5.2.1.2 Capture Regions

For a given shortest path Π^*, its **capture region**, denoted by $C(\Pi^*)$, is defined by:

$$C(\Pi^*) = \{q \in \mathcal{S} : \exists p \in \Pi^*, \quad d(p, q) \leq \frac{\delta}{2}\}. \tag{5.1}$$

The following properties of the capture region play a crucial role in our strategy.

Lemma 5.2.2. *Let Π^* be a shortest path that is being guarded by a pursuer located on the projection of the evader. Then, the evader cannot enter the capture region of Π^* without being captured.*

Proof. Suppose that on its turn the evader crosses $C(\Pi^*)$. That is, the evader moves from e_1 to e_2 to e_3 such that $e_2 \in C(\Pi^*)$ (Fig. 5.4). If $e_2 \in \Pi^*$ then the evader is captured because the pursuer is located on e_2 which is the projection of e_2. Otherwise, let p_1 and p_2 denote the projection of e_1 and e_2 onto Π^* respectively. Since $e_2 \in C(\Pi^*)$, there exists a point $q \in \Pi^*$ such that $d(q, e_2) \leq \frac{\delta}{2}$ (Eq. 5.1). We show that the pursuer can capture the evader on its next move.

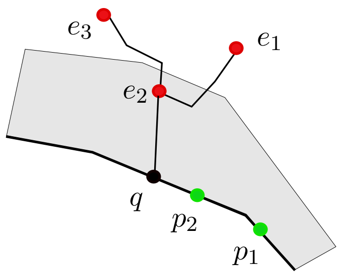


Figure 5.4: $C(\Pi^*)$ is shown in gray. The evader will be captured if it enters $C(\Pi^*)$.

More specifically, we prove that the path $\Pi = \Pi^*(p_1, p_2) + \Pi^*(p_2, q) + \Pi^*(q, e_2) + \Pi^*(e_2, e_3)$ has length less than $1 + \delta$. Thus, the pursuer can move along Π for one unit, and at the end of the move its distance from the evader is at most δ. Therefore, the evader will be captured.

First, p_1, p_2 are the projections of e_1, e_2 respectively. Therefore, $d(p_1, p_2) \leq d(e_1, e_2)$. Next, since p_2 is the projection of e_2 it is closer to q than e_2 is (Section 5.2.1.1). In other words, we have $d(p_2, q) \leq d(e_2, q)$. Recall that $d(e_2, q) \leq \frac{\delta}{2}$ by definition of q . Thus, $d(p_2, q) \leq \frac{\delta}{2}$.

Finally, notice that the length of the evader's path from e_1 to e_3 is at most one. Thus, $d(e_1, e_2) + d(e_2, e_3) \leq 1$. Consequently, we have:

$$\begin{aligned} |\Pi| &= d(p_1, p_2) + d(p_2, q) + d(q, e_2) + d(e_2, e_3) \\ &\leq d(e_1, e_2) + \frac{\delta}{2} + \frac{\delta}{2} + d(e_2, e_3) \leq d(e_1, e_2) + \delta + d(e_2, e_3) \leq 1 + \delta \end{aligned}$$

Thus, the pursuer can take Π and and the end of the move, it can capture the evader. □

Next we show that the capture region is a connected set. As we will see later in Section 5.2.2, this property allows us to analyze our divide-and-conquer approach.

Lemma 5.2.3. *Given a shortest path Π^*, its capture region $C(\Pi^*)$ is connected.*

Proof. Assume to the contrary that $C(\Pi^*)$ is not connected, and let R be one of the connected components of $C(\Pi^*)$. Let q be an arbitrary point in R. According to Eq. 5.1, there exists a point $p \in \Pi^*$ such that $d(p, q) \leq \frac{\delta}{2}$. Let l be a shortest path between p and q. The length of l is $d(p, q)$ which is less than $\frac{\delta}{2}$. Therefore, all of the points on l are in $C(\Pi^*)$. Thus, R is connected to Π^* through l. Consequently, all components of $C(\Pi^*)$ are connected to Π^*. Hence, $C(\Pi^*)$ is a connected set. □

Remark 7. *The capture region of a shortest path is not necessarily a polyhedral subset of $\mathcal{S}$ because the boundary of the capture region can be curved e.g. it can contain circular arcs. In this paper, we assume that we have an oracle that computes the capture region of a given shortest path.*

5.2.1.3 Triangulation into Small Triangles

As we discussed earlier, we will partition $\mathcal{S}$ into small triangles in order to show that the evader will be captured in finite time. Later in Section 5.2.3 we prove that after each phase the pursuers remove at least one of the small triangles from the contaminated region. In the following, we present the algorithm for triangulating the faces into $\frac{\delta}{2}$-small triangles where δ is the capture distance. We also bound the number of triangles.

Definition 4 (Small Triangles)**.** *A triangle is α-small if all of its edges are shorter than α.*

Recall that the input surface is represented as a mesh of triangles. In the following, we show how each of these triangles can be further partitioned into $\frac{\delta}{2}$-small triangles. Suppose that $\triangle abc$ is a triangle which is not $\frac{\delta}{2}$-small.

In the first step, we find one of the bounding rectangles of $\triangle abc$ as follows. In any triangle we have at least two vertices whose angles are less than $\frac{\pi}{2}$. Let b and c be those vertices in $\triangle abc$. We choose the bounding rectangle such that one of its edges is bc (Fig. 5.5). The length of the other edge of the rectangle, which is perpendicular to bc, is the same as the height of the third vertex a with respect to the edge bc (Fig. 5.5).

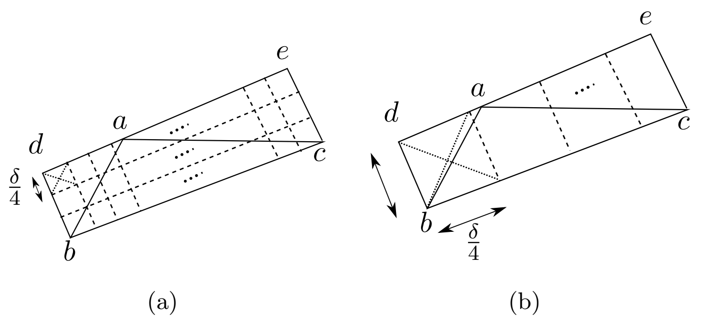


Figure 5.5: Triangulation of $\triangle abc$ into $\frac{\delta}{2}$-small triangles. Case (1) and Case (2) are shown in part (a) and part (b) respectively.

Let us denote the vertices of the bounding rectangle by b, c, e, d as shown in Fig. 5.5(a). There are three cases based on whether the edges of the rectangle have lengths less than $\frac{\delta}{4}$ or not.

Case (1): Both of the edges bc and bd have length greater than $\frac{\delta}{4}$ (Fig. 5.5(a)). In this case, we put a grid of squares of side length $\frac{\delta}{4}$ on the rectangle. Since both bc and bd have length greater than $\frac{\delta}{4}$ we have $O(\frac{|bc|}{\delta}.\frac{|bd|}{\delta})$ square cells. Notice that $O(\frac{|bc|}{\delta}.\frac{|bd|}{\delta}) = O(\frac{A_i}{\delta^2})$ where A_i is the area of $\triangle abc$.

We next partition each square cell into four $\frac{\delta}{4}$-small triangles using its diagonals. In the final step, we consider only the small triangles that are covered by $\triangle abc$. Some of these small triangles might intersect the edges ac and ab. As a result they are divided into smaller triangles, quadrilaterals or pentagons. This is because the edge ac divides a triangle into a triangle and a quadrilateral. If the edge ab intersects the quadrilateral, a pentagon is introduced[1] . We can easily partition the resulting quadrilaterals and pentagons into $O(1)$ $\frac{\delta}{2}$-small triangles by adding diagonals. Therefore, the number of small triangles on $\triangle abc$ is $O(\frac{A_i}{\delta^2})$ where A_i is the area of $\triangle abc$.

Case (2): Exactly one of the edges bc and bd has length greater than $\frac{\delta}{4}$ (Fig. 5.5(b)). In this case, we create a grid of single row with width equal to the longer edge. Using this grid, we can partition $\triangle abc$ into $O(\frac{L_i}{\delta})$ small triangles where L_i denotes the length of the longest edge of $\triangle abc$.

Case (3): Both of the edges bc and bd have lengths less than $\frac{\delta}{4}$. In this case, $\triangle abc$ is already $\frac{\delta}{2}$-small.

We apply this partitioning algorithm to each triangular face on $\mathcal{S}$ which is not $\frac{\delta}{2}$-small.

Lemma 5.2.4. *Using the triangulation algorithm above, the polyhedral surface $\mathcal{S}$ is partitioned into $O(\frac{A}{\delta^2} + \frac{L}{\delta} + N)$ small triangles where A is the area of $\mathcal{S}$, L is the sum of the lengths of the edges on $\mathcal{S}$, N is the number of triangles in the original mesh representation of $\mathcal{S}$, and δ is the capture distance.*

Proof. The surface is initially represented by a mesh of N triangles (see Section 5.1). Each of these triangles falls into one of the three cases of the algorithm above. The three cases introduce $O(\frac{A_i}{\delta^2})$, $O(\frac{L_i}{\delta})$ or $O(1)$ small triangles where A_i is the area and L_i is the length of the longest edge on the corresponding triangle. Notice that $L_i = \max(|a_ib_i|, |a_ic_i|, |b_ic_i|)$ where $\triangle a_ib_ic_i$ is the i^{th} face on $\mathcal{S}$. Therefore,

$$\sum_{\triangle a_ib_ic_i} L_i \leq 2L$$

where

$$L = \sum_{\triangle a_ib_ic_i} (|a_ib_i| + |a_ic_i| + |b_ic_i|)$$

[1] Since ac and ab intersect each other at the endpoint a, the quadrilateral is cut into a triangle and a pentagon and not a 6-gon.

As a result, we have

$$O(\sum_{\triangle a_i b_i c_i} (\frac{A_i}{\delta^2} + \frac{L_i}{\delta} + 1)) = O(\frac{A}{\delta^2} + \frac{L}{\delta} + N)$$

small triangles in total. □

5.2.2 The Three Pursuer Strategy

We are now ready to present the details of our pursuit strategy. Let us denote the paths Π_1, Π_2 and Π_3 guarded at the i^{th} phase by Π_1^i, Π_2^i and Π_3^i respectively. Moreover, let a_j and b_j be the two endpoints of Π_j^i for $j \leq 3$. The pursuer p_j is assigned to guard the path Π_j.

5.2.2.1 Initialization: Phase $i = 1$

We first present the initialization of Π_1 and Π_2. Initially in the first phase $i = 1$, two distinct points in $\mathcal{S}$ are selected as the endpoints of Π_1. Then Π_1 is a shortest path between a_1 and b_1 inside $\mathcal{S}$. If $\partial C(\Pi_1) \setminus \partial \mathcal{S} = \emptyset$ then by Lemma 5.2.14, the evader will be captured by the pursuer p_1. If $\partial C(\Pi_1) \setminus \partial \mathcal{S} \neq \emptyset$, the endpoints of Π_2 are selected as two arbitrary points in $\partial C(\Pi_1) \setminus \partial \mathcal{S}$. See Fig. 5.6. Then, Π_2 is a path that connects a_2 to b_2 and is a shortest path inside $\mathcal{S} \setminus C(\Pi_1)$.

Note that even though the endpoints of Π_2 are selected arbitrarily, Π_2 will not intersect Π_1 since Π_1 is contained inside $C(\Pi_1)$. As we will formalize later, the property that Π_1 does not intersect Π_2 is an invariant that the pursuers try to maintain throughout the game.

Two pursuers p_1 and p_2 guard Π_1 and Π_2 by locating themselves on the projection of the evader onto Π_1 and Π_2 respectively. Therefore, at the end of the first phase, the evader is restricted to lie in the region $\mathcal{S}_1 = \mathcal{S} \setminus (C(\Pi_1) \cup C(\Pi_2))$.

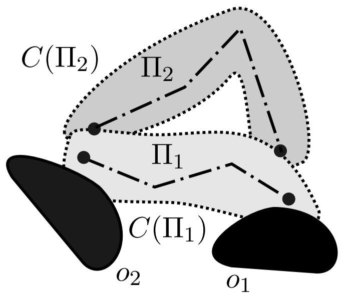


Figure 5.6: Initially, Π_1 is selected as a shortest path between two arbitrary points in $\mathcal{S} \setminus O$. Recall that $\bigcup_{i=1}^{M} \partial O_i \subseteq \partial \mathcal{S}$. The obstacles are shown in black while the capture regions are depicted in gray. The endpoints of Π_2 are chosen as non-boundary points on $\partial C(\Pi_1)$.

Definition 5 (The contaminated region $\mathcal{S}_i$)**.** *The contaminated region $\mathcal{S}_i$ is the connected component of $\mathcal{S}_{i-1} \setminus (C(\Pi_1^i) \cup C(\Pi_2^i))$ that contains the evader at the i^{th} phase.*

The evader cannot exit $\mathcal{S}_i$ without being captured. The region $\mathcal{S}_i$ is bounded by $C(\Pi_1^i)$, $C(\Pi_2^i)$ and $\partial \mathcal{S}$. Notice that initially $\mathcal{S}_0 = \mathcal{S}$.

5.2.2.2 Invariants

By exploiting the properties of capture regions, which we discuss shortly, we will show that the pursuers can maintain the following invariants:

- Invariant (I0): At each phase i, the boundary of the contaminated region $\partial \mathcal{S}_i$ is topologically equivalent to a Jordan curve.
- Invariant (I1): At each phase i, the structure of $\partial \mathcal{S}_i$ falls into one of the following two cases (Lemma 5.2.7). Fig. 5.7 shows an illustration.
  - Case (1): $\partial \mathcal{S}_i$ is composed of a connected non-empty subset of $\partial C(\Pi_{j_1}^i) \cup \partial \mathcal{S}$ and also a connected non-empty subset of $\partial C(\Pi_{j_2}^i) \cup \partial \mathcal{S}$ where $j_1, j_2 \in \{1, 2, 3\}$ and $j_1 \neq j_2$. In other words, both paths contribute to the contaminated region.

  - Case (2): $\partial\mathcal{S}_i$ is composed of a connected non-empty subset of $\partial C(\Pi^i_{j_1}) \cup \partial\mathcal{S}$ ($j_1 \in \{1,2,3\}$). In other words, only one path contributes to the contaminated region.

Remark 8. *From now on, for simplicity of notation we re-label the paths and their guarding pursuers at the beginning of each phase such that $j_1 = 1$ and $j_2 = 2$.*

- Invariant (I2): The paths Π^i_1 and Π^i_2 do not intersect each other.

In Lemma 5.2.5, we present a property of capture regions, which is crucial to maintain the above invariants. We provide its proof in Section 5.2.4. In the lemma, we consider the connected components of $\partial\mathcal{S}_i$ and label them as A, B, C based on whether they are associated with Π_1, Π_2 or $\partial\mathcal{S}$.

Lemma 5.2.5. *Suppose that $\partial\mathcal{S}_i$ is partitioned into a set of connected components $\{A_1, A_2, \cdots, A_{k_1}, B_1, B_2, \cdots, B_{k_2}, C_1, C_2, \cdots, C_{k_3}\}$ such that $A_j \subseteq \partial C(\Pi^i_1) \setminus \partial C(\Pi^i_2)$, $B_j \subseteq \partial C(\Pi^i_2) \setminus \partial C(\Pi^i_1)$, and $C_j \subseteq \partial\mathcal{S}$. Let $L_1 = \bigcup_{j=1}^{j=k_1} A_j$, $L_2 = \bigcup_{j=1}^{j=k_2} B_j$ and $F = \bigcup_{j=1}^{j=k_3} C_j$. Then there exists a partitioning of F into F_1, F_2 such that both $F_1 \cup L_1$ and $F_2 \cup L_2$ are connected.*

Intuitively the lemma says that if $\partial\mathcal{S}_i$ is composed of $\partial\mathcal{S}, \partial C(\Pi^i_1)$ and $\partial C(\Pi^i_2)$, then as we traverse $\partial\mathcal{S}_i$ between two points $q_1, q_2 \in L_1$ (or L_2), we only encounter points that belong to L_1 (respectively L_2). Here, L_j is in fact the contribution of p_j for keeping the evader contained in $\mathcal{S}_i$.

5.2.2.3 Strategy: Phase $i > 1$

By maintaining the above invariants on the structure of $\mathcal{S}_i$ we guarantee that guarding the boundary of $\mathcal{S}_i$ requires at most two pursuers. Consequently, at the end of each phase at least one pursuer is free.

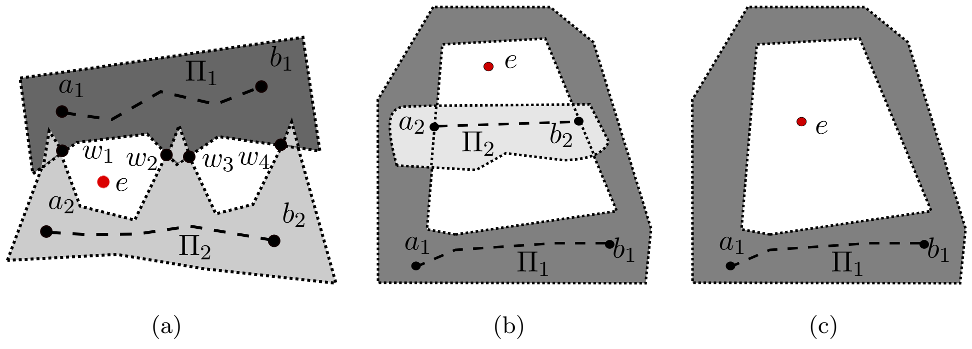


Figure 5.7: The paths Π_1 and Π_2 are shown in dashed lines (For simplicity of the notation, we omit the superscript i in Π^i). The boundary of the capture regions are shown in dots. The capture regions are shaded in gray. (a,b) Two examples for case (1) are illustrated. (c) Case (2) is demonstrated.

The free pursuer p_3 is used for splitting the contaminated region $\mathcal{S}_i$ by guarding Π_3^i. We will choose the path Π_3^i to divide $\mathcal{S}_i$ into two smaller subsets. After placing p_3 on guard position at Π_3^i, the evader cannot cross Π_3^i. The new contaminated region $\mathcal{S}_{i+1}$ is the subset of $\mathcal{S}_i$ that contains the evader defined by either Π_3^i, Π_1^i or Π_3^i, Π_2^i. Therefore, $\mathcal{S}_{i+1}$ is guarded by p_3, p_1 or p_3, p_2. As a result, one of the pursuers p_1 or p_2 is free. Thus, we can repeat splitting $\mathcal{S}_{i+1}$ using the free pursuer. Notice that the pursuers and their corresponding paths are re-labeled so that p_3 is the free pursuer (Remark 8).

Lemma 5.2.6. *Consider the contaminated region $\mathcal{S}_i$ in the i^{th} phase. We can select a path Π_3^i inside $\mathcal{S}_i$ such that:*

- *Π_3^i is a shortest path in $\mathcal{S}_i$ such that it does not intersect Π_1^i and Π_2^i.*
- *Π_3^i splits $\mathcal{S}_i$ into two smaller subsets.*
- *Each of the resulting smaller subsets can be guarded by at most two pursuers.*

Proof. In Lemma 5.2.7 we show that Π_3^i can be selected such that the resulting two subsets can be guarded by only two pursuers. In Lemma 5.2.9 we show that the resulting subsets are strictly smaller than $\mathcal{S}_i$. □

Let us first discuss the selection of the endpoints of the splitting path Π_3^i such that the invariants (I1) and (I2) are maintained throughout the game. Later in Section 5.2.3 we show that the contaminated regions shrinks every time p_3 splits it.

Lemma 5.2.7. *At each phase i, the path Π_3^i can be selected such that it is a shortest path in $\mathcal{S}_i$, and it does not intersect Π_1^i and Π_2^i. Furthermore, Π_3^i is guaranteed to split $\mathcal{S}_i$ into two subsets each of which can be guarded by at most two pursuers.*

Proof. For simplicity of notation we omit the superscript i in Π^i. We prove the claim by induction on the phase number i. In particular, we first assume that the invariants hold for $\mathcal{S}_i$. Then, in a constructive approach we show that for each of the three cases we can select the endpoints of Π_3 such that $\mathcal{S}_{i+1}$ conforms to the invariants as well.

For the induction basis, notice that $\mathcal{S}_1$ is the connected component of $\mathcal{S} \setminus (C(\Pi_1) \cup C(\Pi_2))$ that contains the evader. Therefore, $\partial\mathcal{S}_1$ is composed of $\partial\mathcal{S}$, $\partial C(\Pi_1)$ and $C(\Pi_2)$, and is also homeomorphic to a Jordan curve. Thus according to Lemma 5.2.5, $\partial\mathcal{S}_1$ is composed of a connected component of $\partial C(\Pi_1) \cup \partial\mathcal{S}$ and perhaps a a connected component of $\partial C(\Pi_2) \cup \partial\mathcal{S}$. In other words, $\partial\mathcal{S}_1$ conforms to Invariant (I1). Furthermore, since Π_2 is chosen inside $\mathcal{S} \setminus C(\Pi_1)$, Invariant (I2) is also satisfied for $\mathcal{S}_1$.

Next, for the inductive step, suppose that (I1) holds for $\mathcal{S}_i$. Therefore, $\partial\mathcal{S}_i$ is composed of a connected component of $\partial C(\Pi_1) \cup \partial\mathcal{S}$ and perhaps a connected component of $\partial C(\Pi_2) \cup \partial\mathcal{S}$. Let us denote these components by L_1 and L_2 respectively. First, for each of the two cases in Invariant (I1), we present the selection of a_3, b_3 such that (I1) is satisfied in the next phase for $\mathcal{S}_{i+1}$.

- Case (1): Consider the intersection arcs between L_1 and L_2. From each arc select an arbitrary point and denote them by w_1 and w_2 respectively (Fig. 5.7(a)). Then, we select $\{a_3, b_3\}$ to be the pair $\{w_1, w_2\}$.

- Case (2): The endpoints a_3, b_3 are chosen as two arbitrary distinct points on $\partial\mathcal{S}_i$ (Fig. 5.7(c)).

We next show that (I2) holds for $\mathcal{S}_{i+1}$. First, $\partial\mathcal{S}_i$ is homeomorphic to a Jordan curve. Second, Π_3 is a path inside $\mathcal{S}_i$ between a_3 and b_3. Therefore, Π_3 divides $\mathcal{S}_i$ into two smaller subsets.

Without loss of generality suppose that the evader is in between L_1 and Π_3. Notice that $C(\Pi_3)$ is connected (Lemma 5.2.3). Therefore, the subset of $\mathcal{S}_i$ that contains the evader is bounded by $\partial C(\Pi_1)$, $\partial C(\Pi_3)$ and $\partial \mathcal{S}$. As a result of Lemma 5.2.5, the new subset conforms to Invariant (I1). □

5.2.2.4 Examples

We now present interesting examples for each of the two cases in invariant (I1). An abstract illustration of case (2) is shown in Fig. 5.7(b) and a specific example of this configuration is presented in Fig. 5.8. The example in Fig. 5.8 is the following. Imagine a cone and cut it at height $\frac{\delta}{2}$ from its base and then mount a half-sphere on top of it (Fig. 5.8(a)). The endpoints of Π_1 are chosen as the antipodal points on the base circle. The endpoints of Π_2 are also antipodal on the cutting circle (Fig. 5.8(a)). The perimeter of the base circle, the circle that a_1, b_1 lie on, is chosen small enough such that $C(\Pi_1)$ includes the whole base circle and also the portion of the surface up to height $\frac{\delta}{2}$. Now, observe that if a_2, b_2 are chosen as antipodal points on $\partial \mathcal{S}_i$, there exists a shortest path between them that in on the top of the half-sphere as shown in Fig. 5.8(a). With the choice of the evader location shown in Fig. 5.8(b), which provides a top view of the environment, we will have the configuration illustrated in Fig. 5.7(b).

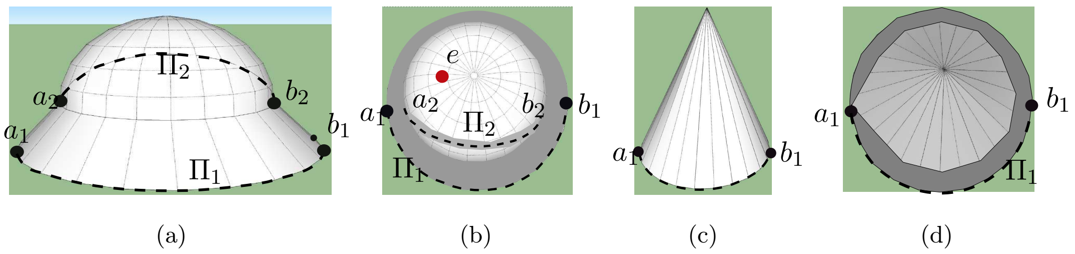


Figure 5.8: The capture regions of the paths Π_1 and Π_2 are shown in gray. The examples for case (2) and case (3) are shown. Notice that in both examples the shortest path Π_1 is on the cone and is obtained as follows. Flatten the cone into a plane. Then Π_1 is a line segment in this plane. (a, b) Side view and top view of an example for case 2 are shown respectively. (c, d) Side view and top view of an example for case 3 are shown.

A similar example for case (3) is shown in Fig. 5.8(c) and Fig. 5.8(d). The example is a cone, and the endpoints of Π_1 are the antipodal points on the base circle of the cone. The perimeter of the base circle is chosen small enough such that the base circle is completely inside $C(\Pi_1)$.

Remark 9. *Notice that for simplicity the examples presented above are not polyhedral. It is not difficult to see that these examples can be approximated by polyhedral surfaces such that the argument above is still valid.*

In the next section, we show that there can only be finitely many phases before the evader is captured.

5.2.3 Making Progress

We now prove that the evader will be captured in finite time. To do so, we provide an upper bound on the number of phases as well as an upper bound on the number of time steps in each phase.

Let us first introduce our notion of progress after each phase. Let Π_3^i be the shortest path inside $\mathcal{S}_i$ that splits the contaminated region $\mathcal{S}_i$. We start by marking all $\frac{\delta}{2}$-small triangles of $\mathcal{S}$ as *contaminated*. At the end of the i^{th} phase, we mark the small triangles that are touched by the splitting path Π_3^i as *cleared*. We show that each small triangle is marked at most once. Thus the number of phases is bounded by the number of $\frac{\delta}{2}$-small triangles on $\mathcal{S}$. We take advantage of the following observation:

Lemma 5.2.8. *The capture region of* Π_3^i *contains all the* $\frac{\delta}{2}$*-small triangles* f *that are being touched by* Π_3^i.

Proof. The distance between any pair of points inside a $\frac{\delta}{2}$-small triangle is at most $\frac{\delta}{2}$. Since Π_3^i and f have a common point, namely q, the distance between q and all the points inside f is at most $\frac{\delta}{2}$. Thus $f \in C(\Pi_3^i)$ (see Definition 5.1). □

Intuitively, after guarding Π_3^i we remove the capture region of Π_3^i from $\mathcal{S}_i$ in order to obtain the new contaminated region $\mathcal{S}_{i+1}$. Therefore, f does not appear in $\mathcal{S}_{i+1}$ and also in the future regions $\mathcal{S}_j, j > i$. Hence, the small triangle f cannot be marked by Π_3^j because Π_3^j marks the faces that it is touching in $\mathcal{S}_j$ and $\mathcal{S}_j$ does not contain f. The following lemma provides a formal proof:

Lemma 5.2.9. *The three pursuer strategy ensures capture after at most* M *phases where* M *is the number of* $\frac{\delta}{2}$*-small triangles on* $\mathcal{S}$.

Proof. The splitting path Π_3^i is computed inside $\mathcal{S}_i$. Therefore:

$$f \text{ is marked clear in the } i^{th} \text{ phase if } \Pi_3^i \cap f \cap \mathcal{S}_i \neq \emptyset \tag{5.2}$$

Clearly, Π_3^i is touching at least one small triangle f. This is because the endpoints of Π_3^i are distinct (otherwise $\mathcal{S}_i = \emptyset$ which means that the evader is already captured). In the following, we show that each f is marked clear at most once. Assume the contrary and suppose that a small triangle f is marked twice: at phases i and j where $i < j$. According to Definition 5 we have $\mathcal{S}_{i+1} \cap C(\Pi_3^i) = \emptyset$. Together with the observation that $f \subseteq C(\Pi_3^i)$, as in Lemma 5.2.8, we must have $f \cap \mathcal{S}_{i+1} = \emptyset$. According to our assumption, f is marked at the j^{th} phase as well. Thus, according to (5.2) we must have $f \cap \mathcal{S}_j \neq \emptyset$.

Now observe that $\mathcal{S}_{k+1} \subset \mathcal{S}_k$ for all k since $\mathcal{S}_{k+1}$ is obtained by removing $C(\Pi_3^k)$ from $\mathcal{S}_k$. Therefore, we have $\mathcal{S}_j \subset \mathcal{S}_i$ for $j > i$. Thus, $\mathcal{S}_j \subseteq \mathcal{S}_{i+1}$, and also we have $f \cap \mathcal{S}_j \subseteq \mathcal{S}_j \subseteq \mathcal{S}_{i+1}$. Therefore, $f \cap \mathcal{S}_j \subseteq \mathcal{S}_{i+1}$. Moreover, $f \cap \mathcal{S}_j \subseteq f$. Consequently, $f \cap \mathcal{S}_j \subseteq f \cap \mathcal{S}_{i+1}$. But, $f \cap \mathcal{S}_{i+1} = \emptyset$ and $f \cap \mathcal{S}_j \neq \emptyset$ which is a contradiction. Hence, each small triangle is marked at most once. Therefore, the number of phases is bounded by the number of $\frac{\delta}{2}$-small triangles on $\mathcal{S}$. □

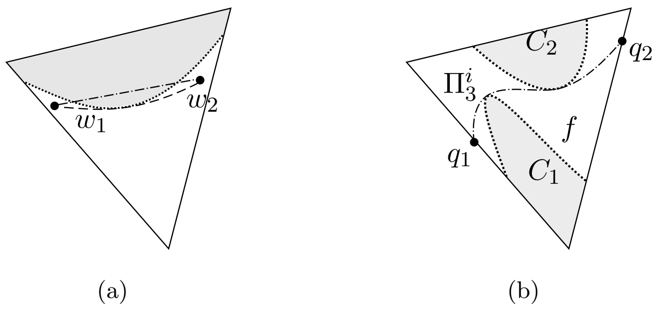


Figure 5.9: (a) The shortest path between w_1 and w_2 in $\mathcal{S}$ is a line segment. When we remove the gray region from $\mathcal{S}$, the shortest path in the new region is longer than the original shortest path. (b) In order to catch up with the projection, the pursuer moves along the segment q_1q_2.

Next, we provide an upper bound on the number of time-steps in each phase. This time is required for p_3 to catch up with the evader's projection. The pursuer's strategy is to walk along Π_3^i until it hits the evader's projection. We show that we need $O(M\frac{\delta}{2})$ steps where M is total number of small triangles on $\mathcal{S}$. First, notice that the number of steps needed cannot be bounded in terms of the length of Π_3^i. This is because Π_3^i is computed in $\mathcal{S}_i$ and not $\mathcal{S}$, and the shortest path between two points in $\mathcal{S}_i$ can be longer than their shortest path in $\mathcal{S}$ (Fig. 5.9(a)).

The pursuer's strategy is as follows. Let f be a small triangle on Π_3^i. Moreover, let q_1 and q_2 be the endpoints of $\Pi_3^i \cap f$ (see Fig. 5.9(b)). Instead of moving along $\Pi_3^i(q_1, q_2)$ the pursuer takes a shortcut and simply moves along the segment q_1q_2. Since Π_3^i is within the capture distance of the pursuer, the evader cannot cross $\Pi_3^i(q_1, q_2)$. As a result, the pursuer can protect Π_3^i and meanwhile travel a shorter distance. Furthermore, Π_3^i enters each small triangle at most once. Thus, the number of steps is at most $O(M\frac{\delta}{2})$.

We are now ready to prove our result which was given in Theorem 5.2.1.

Proof of Theorem 5.2.1 Let M be the total number of $\frac{\delta}{2}$-small triangles on $\mathcal{S}$. There are at most M phases (Lemma 5.2.9). Each phase takes at most $O(M\frac{\delta}{2})$ steps. Thus, the capture time is $O(M^2\frac{\delta}{2})$. According to Lemma 5.2.4 we have $M = O(\frac{A}{\delta^2} + \frac{L}{\delta} + N)$ where A is the area of $\mathcal{S}$, L is the sum of the length of the edges on $\mathcal{S}$, and N is the number of triangles in the original triangular mesh representation of $\mathcal{S}$. Therefore, the capture time is $O((\frac{A}{\delta^2} + \frac{L}{\delta} + N)^2\frac{\delta}{2})$. $\square$

Finally, in the next section we present proofs of the lemmas we saw throughout the section as well as auxiliary lemmas that are needed.

5.2.4 Correctness Proofs

In this section, we present the proofs of our lemmas to show the correctness of our proposed strategy. We start by proving Lemma 5.2.5. We then continue by presenting some auxiliary lemmas that are required in our proofs.

Proof of Lemma 5.2.5. For simplicity of notation, let us drop the superscript i in Π_1^i, Π_2^i and denote them by Π_1 and Π_2 respectively. Notice that $\partial\mathcal{S}_i$ is homeomorphic to a Jordan curve.

Assume to the contrary that for any choice of F_1, F_2 either $L_1 \cup F_1$ or $L_2 \cup F_2$ is disconnected. Observe that $\partial\mathcal{S}_i$ is homeomorphic to a Jordan curve, and L_1, L_2, F partition $\partial\mathcal{S}_i$. Therefore, according to Lemma 5.2.12 there are points $q_1, q_2 \in L_1$ and

$w_1, w_2 \in L_2$ that are arranged as q_1, w_1, q_2, w_2 in clockwise or counter clockwise order around $\partial\mathcal{S}_i$.

Main idea: The main idea of our proof is the following: First, using q_1, q_2, Π_1 and a portion of $\partial\mathcal{S}_i$ we construct a Jordan curve. Then using w_1, w_2 we show the existence of a point in the interior of $\partial\mathcal{S}_i$ and a point in the exterior of $\partial\mathcal{S}_i$ which results in the existence of a path in C_1 which intersects a path in C_2. Therefore, according to Lemma 5.2.10 we conclude that $q_1 \in C_2$ or $q_2 \in C_2$ or $w_1 \in C_1$ or $w_2 \in C_1$ which contradicts the assumption that $q_1, q_2 \in C_2 \setminus C_1$ and $w_1, w_2 \in C_1 \setminus C_2$

Construction of a Jordan Curve: Let us begin by noticing that since $q_1 \in C_1$ there exists a point on Π_1 namely s_q^1 such that $d(s_q^1, q_1) \leq \frac{\delta}{2}$. Similarly, there exists a point s_q^2 on Π_1 such that $d(s_q^2, q_2) \leq \frac{\delta}{2}$. Likewise, define $s_w^1, s_w^2 \in \Pi_2$ such that $d(s_w^1, w_1), d(s_w^2, w_2) \leq \frac{\delta}{2}$. Consider $\Pi(s_q^1, q_1)$ and $\Pi(s_q^2, q_2)$ and observe the following properties:

1. Both $\Pi(s_q^1, q_1)$ and $\Pi(s_q^2, q_2)$ are **not** self-intersecting. Because otherwise they could be shortened.

2. Both $\Pi(s_q^1, q_1)$ and $\Pi(s_q^2, q_2)$ do not intersect Π_1. Because otherwise q_1, q_2 would belong to the interior of C_1 and not ∂C_1 (recall that $q_1, q_2 \in \partial\mathcal{S}_i$).

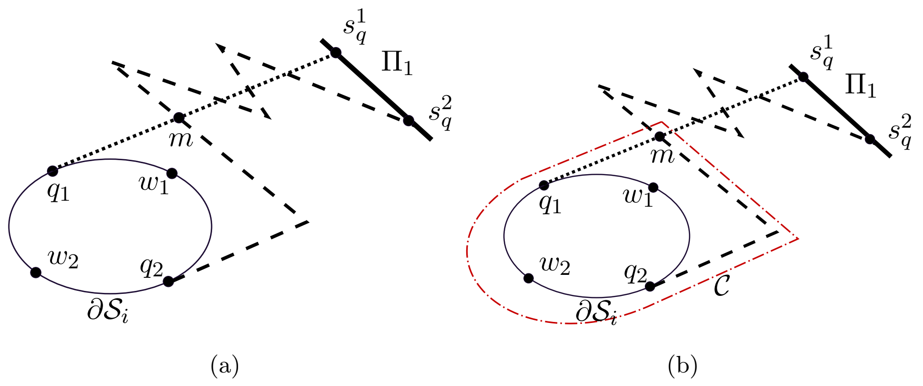


Figure 5.10: Construction of the Jordan curve $\mathcal{C}$ is shown. (a) The case when $\Pi(s_q^1, q_1)$ and $\Pi(s_q^2, q_2)$ intersect each other is demonstrated. (b) $\mathcal{C}$ is shown in the aforementioned case.

Now we construct a Jordan curve by concatenating sub-paths of $\partial\mathcal{S}_i, \Pi_1$, $\Pi(s_q^1, q_1)$ and $\Pi(s_q^2, q_2)$. Let us denote this Jordan curve by $\mathcal{C}$ (Fig. 5.10(b)). Since $\partial\mathcal{S}_i$ is a Jordan curve, it divides $\mathcal{S}$ into two sets: the **interior** and the **exterior** of $\partial\mathcal{S}_i$. All the paths Π_1, Π_2, $\Pi(s_q^1, q_1)$ and $\Pi(s_q^2, q_2)$ are in the exterior of $\partial\mathcal{S}_i$. There are two cases whether $\Pi(s_q^1, q_1)$ intersect $\Pi(s_q^2, q_2)$ or not:

1. $\Pi(s_q^1, q_1)$ does not intersect $\Pi(s_q^2, q_2)$: In this case, construct $\mathcal{C}$ by concatenating $\Pi(q_1, s_q^1), \Pi(s_q^1, s_q^2), \Pi(s_q^2, q_2)$ and $\partial\mathcal{S}_i(q_1, q_2)$ where $\partial\mathcal{S}_i(q_1, q_2)$ is chosen arbitrarily among the two sub-paths between q_1, q_2 along $\partial\mathcal{S}_i$.

2. $\Pi(s_q^1, q_1)$ intersects $\Pi(s_q^2, q_2)$: Let m be the intersection point which is closest to q_1 along $\Pi(s_q^1, q_1)$. See Fig. 5.10(a). Then construct $\mathcal{C}$ by concatenating $\Pi(q_1, m)$, $\Pi(m, q_2)$ and $\partial\mathcal{S}_i(q_1, q_2)$ where $\partial\mathcal{S}_i(q_1, q_2)$ is chosen arbitrarily among the two sub-paths between q_1, q_2 along $\partial\mathcal{S}_i$. See Fig. 5.10(b).

Contradiction: Finally, we construct a path between w_1 and w_2 that intersects $\mathcal{C}$ as a result of the Jordan curve theorem. As a corollary of the intersection we will show a contradiction by Lemma 5.2.10 and Lemma 5.2.11.

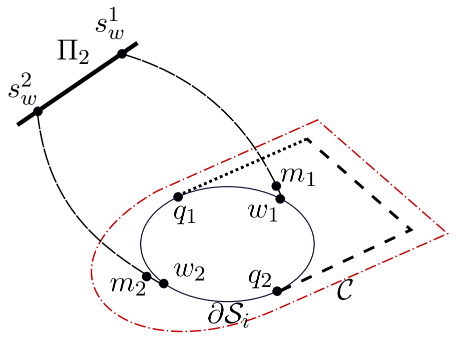


Figure 5.11: The path $\Pi = \Pi(w_1, s_w^1) + \Pi_2(s_w^1, s_w^2) + \Pi(s_w^2, w_2)$ intersects the Jordan curve $\mathcal{C}$.

Let us begin by constructing a path $\Pi = \Pi(w_1, s_w^1) + \Pi_2(s_w^1, s_w^2) + \Pi(s_w^2, w_2)$. See Fig. 5.11. Then there exist two points m_1, m_2 on Π one of which is in the neighborhood of w_1 and the other is in the neighborhood of w_2 such that either:

$$m_1 \in \text{interior}(\mathcal{C}) \land m_2 \in \text{exterior}(\mathcal{C}) \tag{5.3}$$

or:

$$m_1 \in \text{exterior}(\mathcal{C}) \wedge m_2 \in \text{interior}(\mathcal{C}) \tag{5.4}$$

Therefore $\Pi(m_1, m_2)$ intersects $\mathcal{C}$ (by Jordan curve theorem). Let h be the intersection point. There are two cases:

1. $h \notin \Pi_1 \cup \Pi_2$: According to Lemma 5.2.10 either $q_1 \in C_2$ or $q_2 \in C_2$ or $w_1 \in C_1$ or $w_2 \in C_1$. But this is a contradiction because $q_1, q_2 \in C_1 \setminus C_2$ and $w_1, w_2 \in C_2 \setminus C_1$.

2. $h \in \Pi_1$ or $h \in \Pi_2$: Without loss of generality suppose that $h \in \Pi_2$. Since Π_1 and Π_2 are non-intersecting it must be that $h \in \Pi(q_1, s_q^1)$ or $h \in \Pi(q_2, s_q^2)$. Without loss of generality suppose that $h \in \Pi(q_1, s_q^1)$. According to Lemma 5.2.11 we have $q_1 \in C_2$ which contradicts the initial assumption that $q_1 \in C_1 \setminus C_2$.

Both cases result in contradiction which proves our lemma. □

Lemma 5.2.10. *Let C_1 and C_2 be the capture regions of paths Π_1 and Π_2 respectively. If for a point $p_1 \in C_1$ and a point $p_2 \in C_2$, a shortest path between Π_1 and p_1, and a shortest path between Π_2 and p_2 intersect, then it must be that $p_1 \in C_2$ or $p_2 \in C_1$.*

Proof. Let $\Pi(s_1, p_1)$ and $\Pi(s_2, p_2)$ be the corresponding shortest paths: $\Pi(s_1, p_1)$ is between $s_1 \in \Pi_1$ and p_1, and $\Pi(s_2, p_2)$ is between $s_2 \in \Pi_2$ and p_2. Since p_1 is inside the capture region of Π_1, it must be that $d(s_1, p_1) \leq \frac{\delta}{2}$ (Definition 5.1). Similarly, we have $d(s_2, p_2) \leq \frac{\delta}{2}$. Let q be an intersection point between $\Pi(s_1, p_1)$ and $\Pi(s_2, p_2)$.

The main idea of our proof is the following. The intersection point q divides each of the shortest paths $\Pi(s_1, p_1)$ and $\Pi(s_2, p_2)$ into two sub-paths. We will concatenate a sub-path from one shortest path to a sub-path from the second one and construct new paths of length at most $\frac{\delta}{2}$ between p_1 and Π_2 and similarly between p_2 and Π_1. Therefore, it can be concluded that $p_1 \in C_2$ and $p_2 \in C_1$.

We now start our proof. Consider the sub-path of $\Pi(s_2, p_2)$ from q to p_2, i.e. $\Pi(q, p_2)$, and the sub-path of $\Pi(s_1, p_1)$ from q to p_1 i.e., $\Pi(q, p_1)$. Regarding the length of these two sub-paths, there are two cases:

1. $|\Pi(q, p_2)| \leq |\Pi(q, p_1)|$: Adding $|\Pi(s_1, q)|$ to both sides we get $|\Pi(s_1, q)| + |\Pi(q, p_2)| \leq |\Pi(s_1, q)| + |\Pi(q, p_1)|$. Thus $|\Pi(s_1, p_2)| \leq |\Pi(s_1, p_1)|$. Since $|\Pi(s_1, p_1)| = d(s_1, p_1) \leq \frac{\delta}{2}$ we have $|\Pi(s_1, p_2)| \leq \frac{\delta}{2}$. Furthermore, since $s_1 \in \Pi_1$ we have $p_2 \in C_1$.

2. $|\Pi(q,p_2)| > |\Pi(q,p_1)|$: Adding $|\Pi(s_2,q)|$ to both sides we get $|\Pi(s_2,q)|+|\Pi(q,p_2)| > |\Pi(s_2,q)|+|\Pi(q,p_1)|$. Thus $|\Pi(s_2,p_2)| > |\Pi(s_2,p_1)|$. Since $|\Pi(s_2,p_2)| = d(s_2,p_2) \leq \frac{\delta}{2}$ we conclude $|\Pi(s_2,p_1)| \leq \frac{\delta}{2}$. Since $s_2 \in \Pi_2$ we have $p_1 \in C_2$.

□

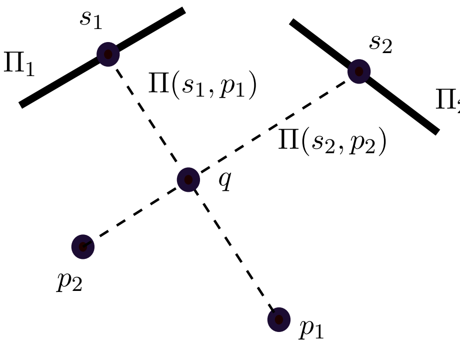


Figure 5.12: Proof of Lemma 5.2.10 is illustrated. A shortest path between $s_1 \in \Pi_1$ and $p_1 \in C_1$ intersects a shortest path between $s_2 \in \Pi_2$ and $p_2 \in C_2$.

Lemma 5.2.11. *Let C_1 and C_2 be the capture regions of paths Π_1 and Π_2 respectively. If for a point $p_1 \in C_1$, a shortest path between Π_1 and p_1 intersects Π_2, then it must be that $p_1 \in C_2$.*

Proof. Let $\Pi(s_1,p_1)$ be the corresponding shortest path: $\Pi(s_1,p_1)$ is between $s_1 \in \Pi_1$ and p_1. Let h be an intersection point between $\Pi(s_1,p_1)$ and Π_2. Consider the subpath of $\Pi(s_1,p_1)$ from h to p_1. Since $p_1 \in C_1$ we have $d(s_1,p_1) \leq \frac{\delta}{2}$ (Definition 5.1). Therefore $d(h,p_1) \leq \frac{\delta}{2}$. Together with the fact that $h \in \Pi_2$ we conclude that $p_1 \in C_2$.

□

Lemma 5.2.12. *Let $\{L_1, L_2, F\}$ be a partitioning of a Jordan curve $\mathcal{C}$ into three sets with the extra condition that F can be the empty set, i.e., $L_1, L_2 \neq \emptyset$ and $F \cup L_1 \cup L_2 = \mathcal{C}$ and $F \cap L_1 = \emptyset, F \cap L_2 = \emptyset, L_1 \cap L_2 = \emptyset$. Let F_1, F_2 be any partitioning of the set F into two (possibly empty) sets. If for any choice of F_1, F_2 either $L_1 \cup F_1$ or $L_2 \cup F_2$ is disconnected then there are points $q_1, q_2 \in L_1$ and $w_1, w_2 \in L_2$ that appear as q_1, w_1, q_2, w_2 in clockwise or counter clockwise order around $\mathcal{C}$.*

Proof. Consider the particular choice of F_1, F_2 where $F_1 = F$ and $F_2 = \emptyset$. According to Lemma 5.2.13, since $\mathcal{C}$ is a Jordan curve, and either $L_1 \cup F_1 = L_1 \cup F$ or $L_2 \cup F_2 = L_2$ is disconnected, there are two non-empty disconnected components of $L_1 \cup F$, namely V_1, V_2 and two non-empty disconnected components of L_2, namely W_1, W_2, such that $V_j, W_j, j \leq 2$ are arranged as V_1, W_1, V_2, W_2 in clockwise or counter clockwise order. The take-away conclusion here is that $W_1, W_2 \subset L_2$, $W_1, W_2 \neq \emptyset$ and that W_1, W_2 are disconnected components of L_2. Similarly, with the choice $F_1 = \emptyset, F_2 = F$, we can show the existence of nonempty connected components of L_1 namely Q_1, Q_2.

We now prove the claim of the lemma. There are two cases regarding the arrangement of Q_1, Q_2, W_1, W_2:

1. With proper labeling Q_i, W_j are arranged as Q_1, W_1, Q_2, W_2 in clockwise order around $\mathcal{C}$: Then any pair of points $q_1 \in Q_1, q_2 \in Q_2$ and any pair of points $w_1 \in W_1, w_2 \in W_2$ will establish the arrangement q_1, w_1, q_2, w_2.

2. With proper labeling Q_i, W_j are arranged as Q_1, Q_2, W_1, W_2 in clockwise order around $\mathcal{C}$: Assume to the contrary that such q_i, w_i do not exist. Therefore, as we a point q along $\mathcal{C}$ from Q_1 to Q_2 as shown in Fig. 5.13 $q \in F$. Similarly, as we move a point q along $\mathcal{C}$ from W_1 to W_2 as shown in Fig. 5.13 $q \in F$.

   Now, move a point p_1 along $\mathcal{C}$ from Q_1 to W_1 as in Fig. 5.13. In the following we enumerate all the possibilities of whether $p_1 \in L_1$ or $p_1 \in L_2$ or $p_1 \in F$. We use an arrow sign ($\rightarrow$) to show the transition from one set to the other e.g., $L_1 \rightarrow L_2$ means that as we move p_1 first $p_1 \in L_1$ and then at some point $p_1 \in L_2$:

$$\begin{aligned} (L_1 \cup F) &\rightarrow (L_2 \cup F) \\ L_2 &\rightarrow (L_2 \cup F) \\ (L_1 \cup F) &\rightarrow (L_1) \\ L_2 &\rightarrow L_1 \end{aligned} \tag{5.5}$$

   Similarly, move a point p_2 along $\mathcal{C}$ from Q_2 to W_2 as shown in Fig. 5.13. The enumeration of all possibilities whether $p_2 \in L_1$ or $p_2 \in L_2$ or $p_2 \in F$ is the same as (5.5). Since for any choice of F_1, F_2 either $L_1 \cup F_1$ or $L_2 \cup F_2$ is disconnected then at least one of p_1, p_2 has to fall into the fourth case in (5.5). Either case, the points q_1, q_2, w_1, w_2 can be selected with the desired arrangement.

□

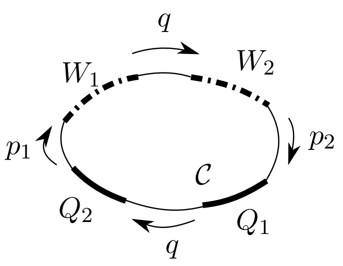


Figure 5.13: Proof of Lemma 5.2.12.

Lemma 5.2.13. *Let $\{L_1, L_2\}$ be a partitioning of a given Jordan curve $\mathcal{C}$ into two sets*[2] *. An example is shown in Fig. 5.14(a). If either of L_1, L_2 is disconnected then there are non-empty disconnected components $Q_1, Q_2 \subset L_1$ and $W_1, W_2 \subset L_2$, that are arranged as Q_1, W_1, Q_2, W_2 in clockwise or counter-clockwise order. See Fig. 5.14(b).*

Proof. Without loss of generality suppose that L_1 is disconnected. Let Q_1, Q_2 be two disconnected components of L_1. Let q denote any point on $\mathcal{C}$. Move q along $\mathcal{C}$ in clockwise direction. Since Q_1, Q_2 are disconnected components of L_1, at some point $q \in L_2$. Let q_1 denote this point (Fig. 5.14(c)). Similarly, as we move q in counter-clockwise direction, at some point $q \in L_2$. Let q_2 denote this point. Both q_1, q_2 belong to L_2 and furthermore they are disconnected by Q_1, Q_2. Therefore, there are two disconnected components of L_2 namely W_1, W_2 such that $q_1 \in W_1$ and $q_2 \in W_2$. Finally, Q_1, Q_2 and W_1, W_2 establish the desired arrangement in Fig. 5.14(b). □

[2] *Note that $L_1, L_2 \neq \emptyset$ since $\{L_1, L_2\}$ is a partitioning. Moreover, each L_i is a collection of sub-paths of $\mathcal{C}$ where each sub-path could be a single point.*

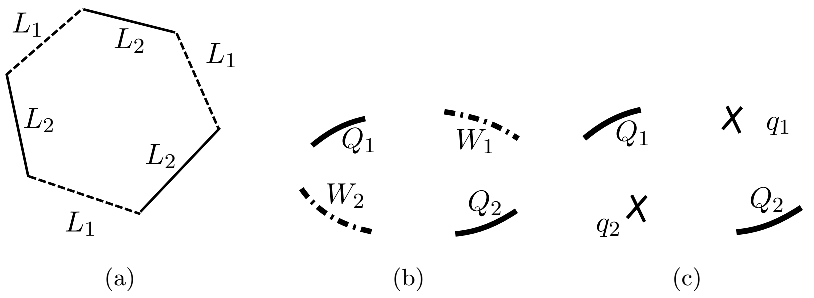


Figure 5.14: (a) A partitioning of the closed curve $\mathcal{C}$ into two sets L_1, L_2 is depicted. Here, both L_1, L_2 have three connected components. (b) If either L_1 or L_2 is disconnected, then there are disconnected components of L_1, L_2 that are arranged as shown in this figure. (c) Since Q_1, Q_2 are disconnected components of L_1, there exist disconnected components of L_2 namely X, Y in between them.

Lemma 5.2.14. *Let C denote a capture region in $\mathcal{S}$. If $\partial C \setminus \partial \mathcal{S} = \emptyset$ then $\mathcal{S} \subseteq C$.*

Proof. Since $\partial C \setminus \partial \mathcal{S} = \emptyset$ then $\partial C \subseteq \partial \mathcal{S}$. If $\partial C = \emptyset$ then we have $\mathcal{S} \subseteq C$. If $\partial C \neq \emptyset$ then $\mathcal{S} \subseteq C$ since both C and $\mathcal{S}$ are connected. □

5.2.5 Summary

So far, we have studied the lion and man game on the surface of a polyhedron with obstacles. The surface is homeomorphic to a sphere or a disk that can contain holes (handles are not allowed). We showed the existence of a capture strategy with three lions when the capture distance is non-zero. In order to compute the strategy on a given polyhedron a subroutine which computes the capture region of a shortest path is needed. This region is then removed from the environment and the computation proceeds iteratively.

Next, we focus on a smaller class of polyhedral surfaces, i.e., convex height-maps which we refer to as convex terrains.

5.3 Single Pursuer on Convex Terrains

In this section, we study the lion and man game on surfaces of convex terrains[3]. We show that the lion can capture the man in finite number of steps determined by the terrain geometry.

A terrain is obtained by assigning a single height value to each point in a bounded region of a plane in $\mathbb{R}^2$. In particular, a terrain is a polyhedral surface with boundary in $\mathbb{R}^3$ such that each of its vertices has a single height value associated with it, i.e. terrains are height maps [79]. To make the presentation easier, we assume that all terrain vertices are at different heights which is attainable by slightly perturbing the height function. A convex terrain is a terrain with a convex height function.

Our pursuit strategy is based on guarding *wavefronts* and pushing them towards the evader. A wavefront at height z is defined as the set of points on the terrain that are on the same height z. We first discretize the terrain by a set of wavefronts. The pursuer starts from the highest wavefront and pushes the frontier wavefront downwards while preventing the evader from entering any previously guarded wavefront. Intuitively, the perimeter of the frontier wavefront is increased in this downward sweep. This allows the pursuer to use the difference between the perimeter of two consecutive wavefronts in order to make progress.

In Section 5.3.1 we present the key concepts we use throughout the section. An overview of the proposed strategy is presented in Section 5.3.2. The discretization of the terrain into wavefronts is explained in Section 5.3.3. Details of the pursuer strategy for guarding the current wavefront and making progress to the next wavefront are presented in Section 5.3.4 and Section 5.3.5 respectively. We present the detailed proof of our lemmas in Section 5.3.6.

5.3.1 Key Concepts: Wavefront, Projection and Image

Let us begin by presenting some important concepts that we will use in our strategy. We refer to the two-dimensional plane with the lowest height, i.e. the $z = 0$ plane, as the *base plane.* We occasionally refer to this plane as the XY-plane. Moreover, we use the coordinate frame XYZ with its origin placed on any arbitrary point in the base

[3] The material in this section appears in [16].

XY-plane (Fig. 5.15).

Definition 6 (The Perpendicular Image and Pre-Image)**.** *For a point $p = (x, y, z)$ on $\mathcal{S}$, the point $q = (x, y)$ is called the* perpendicular image of p *onto the base plane. Also, p is called the* pre-image *of q. Similarly, one can define the image (pre-image) of a path on $\mathcal{S}$ (on the XY-plane).*

Note that we reserve the term *projection* for an important ingredient of our strategy which we will present shortly. We have the following useful proposition.

Proposition 5.3.1. *The pre-image of any continuous path in the XY-plane is a continuous path on $\mathcal{S}$.*

Definition 7. *Let p_1 and p_2 be two distinct points which are on the same face f of $\mathcal{S}$. Consider the straight line segment that connects them on $\mathcal{S}$, and denote its length by L. Also, let l be the length of its perpendicular image. We refer to the ratio $\alpha(p_1, p_2) = \frac{l}{L}$ as the length coefficient associated with p_1 and p_2.*

The length coefficient $\alpha(p_1, p_2)$ is in fact the cosine of the angle between the segment p_1p_2 and the XY-plane. Notice that the largest possible value of this angle is the angle between the face f and the XY-plane. Therefore, the minimum length coefficient is well defined in the sense that it is a finite positive number. This is because faces with vertical edges are not allowed, and also the two points p_1 and p_2 are distinct.

Proposition 5.3.2. *The length coefficients are positive and less than or equal to one, i.e. $0 < \alpha(p_1, p_2) \leq 1$. We refer to the minimum possible length coefficient on $\mathcal{S}$ as $\alpha = \min_{f \in \mathcal{S}} \min_{p_1, p_2 \in f} \alpha(p_1, p_2)$.*

Lemma 5.3.3. *Let p_1 and p_2 be two distinct points on $\mathcal{S}$. Let s be the shortest path between p_1 and p_2 on $\mathcal{S}$. Denote the length of s and its image by $a_{\mathcal{S}}$ and a respectively. Then $a_{\mathcal{S}} \leq \frac{a}{\alpha}$.*

Proof. Let $f_1, f_2, \cdots, f_k$ be the sequence of faces that s passes through. Observe that the portion of s which is on f_i is a line segment (because otherwise, s can be shortened by taking the line segment between the entry and the exit points of f_i). Let s_i denote the line segment on f_i. Denote the length of s_i and its image by $a_{\mathcal{S},i}$ and a_i respectively. Then, $a_{\mathcal{S}} = \sum_i a_{\mathcal{S},i}$ and $a = \sum_i a_i$. By proposition 5.3.2, we have $a_{\mathcal{S},i} \leq \frac{a_i}{\alpha}$. Therefore, $\sum_i a_{\mathcal{S},i} \leq \sum_i \frac{a_i}{\alpha}$. Thus, $a_{\mathcal{S}} \leq \frac{a}{\alpha}$. $\square$

We next present an important ingredient of our strategy: the wavefronts.

Definition 8 (Wavefronts)**.** *We refer to the set of points on $\mathcal{S}$ which are on the same height z as the* wavefront *at z. Throughout the paper, we reserve the letter W for the wavefronts.*

Observe that the wavefront at height z is the intersection of $\mathcal{S}$ with the plane $Z = z$ which are both convex sets. Therefore, wavefronts are also convex polygons. Also, the image of a wavefront in the XY-plane is obtained by taking the perpendicular image of every point of the wavefront.

Definition 9. *Let $p_1, p_2 \in W$ be two points on the wavefront W. We denote the shorter path from p_1 to p_2 along W by $W(p_1, p_2)$, and its length by $d_W(p_1, p_2)$. We also denote the length of the segment $p_1^i p_2^i$ in the XY-plane by $d_{XY}(p_1, p_2)$.*

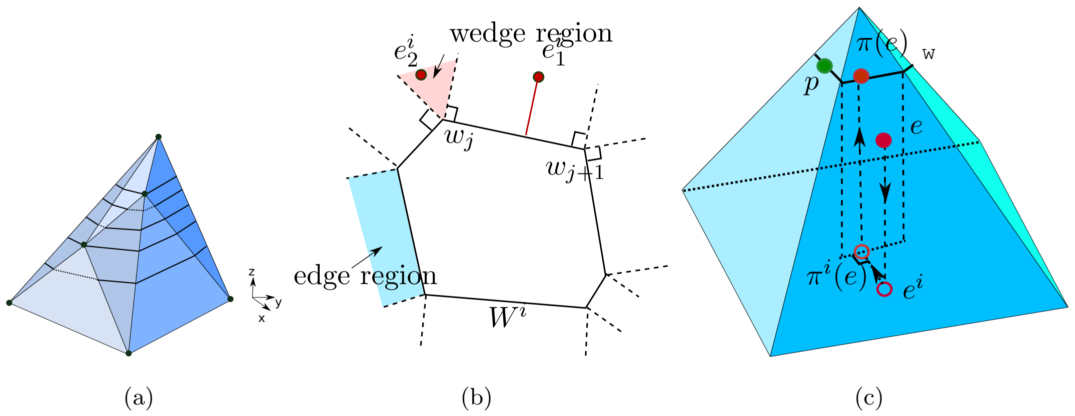


Figure 5.15: (a) Discretization of $\mathcal{S}$ by a set of wavefronts. (b) The partitioning of the exterior of W^i into wedge regions and edge regions. (c) The projection of e onto the wavefront W. Here, p_1 and p_2 are the projections of e_1 and e_2 respectively.

Let W^i be the perpendicular image of a wavefront W. We partition the region outside W^i (in the base plane) into regions of two types: the *edge regions* and the *wedge regions* as follows. Suppose that the vertices of W^i are labeled as $\{w_1, w_2, ..., w_n\}$ in the clockwise order. See Fig. 5.15(b) for an illustration. Let l_j^1 and l_j^2 be the two perpendicular lines to edges $w_{j-1}w_j$ and $w_j w_{j+1}$ which are drawn from w_j.

Definition 10 (Wedge Regions and Edge Regions)**.** *For an edge $w_j w_{j+1}$, its corresponding edge region is the region in between l_j^2 and l_{j+1}^1. For a vertex $w_j \in W^i$, its corresponding wedge region is the region in between l_j^1 and l_j^2. Notice that these regions are non-overlapping since W is a convex polygon.*

We use the following feature of $\mathcal{S}$ to provide the capture time of our strategy:

Definition 11 (Wedge Angle)**.** *For a given wavefront vertex w_j, the* wedge angle *is defined as the angle between l_j^1 and l_j^2 in the base plane.*

We are now ready for presenting the key concept in guarding the wavefronts: the *projection* of the evader onto a wavefront. Let e^i and W^i be the perpendicular images of e and a wavefront W onto the XY-plane respectively. Also, suppose that e^i is outside the region enclosed by W^i. The *projection* of e onto the wavefront W is defined as follows.

Definition 12 (Projection onto a Wavefront)**.** *Consider the partitioning of the exterior region of W into wedge regions and edge regions. See Fig. 5.15(b). There will be two cases based on the location of e^i: 1) e^i is inside the edge region associated with an edge $w_j w_{j+1}$ (e.g. e_1^i); 2) e^i is inside the wedge region of a vertex w_j (e.g. e_2^i). In the first case, let p denote the intersection of the edge $w_j w_{j+1}$ and the perpendicular line to the edge $w_j w_{j+1}$ which passes through e^i. In the second case, let p denote the vertex w_j. Then, the projection of e onto W is the pre-image of p on $\mathcal{S}$ (Fig. 5.15(c)). We denote this point on $\mathcal{S}$ as $\pi(e, W)$.*

Remark 1. *Notice that the perpendicular image in Definition 6 is different from the projection onto a wavefront in Definition 12. For a point $p \in \mathcal{S}$, its image is denoted by p^i while its projection onto W is denoted by $\pi(p, W)$.*

5.3.2 Overview of the Pursuit Strategy

The idea of our pursuit strategy is the following. We first discretize the surface of $\mathcal{S}$ by a set of wavefronts (Section 5.3.3). Initially, the pursuer goes to the highest point of $\mathcal{S}$. (We assume that this point is unique. It is not too difficult to show that our strategy is applicable if there are more than one point with the same height.). The highest point is in fact the first wavefront in the set of wavefronts. The pursuer's strategy has two components: (1) guarding the current wavefront in order to prevent the evader from crossing it without being captured, (2) making progress by moving to the next wavefront.

Definition 13. *We refer to the wavefront that the pursuer is currently guarding as the* frontier *wavefront. Throughout the paper, we denote the frontier wavefront by W and the wavefront right below it by W_n.*

We achieve these two goals as follows. We consider the images of the wavefronts in the XY-plane (Fig. 5.17(b)). Meanwhile we use the images of the players and projection of the evader onto wavefronts to transform the game to the base plane and guide the pursuer's strategy on $\mathcal{S}$ (Fig. 5.15(c)). In order to guard the current wavefront W, the pursuer uses the projection of the evader onto W because the projection has the nice property that it is closer to all points on W than the evader.

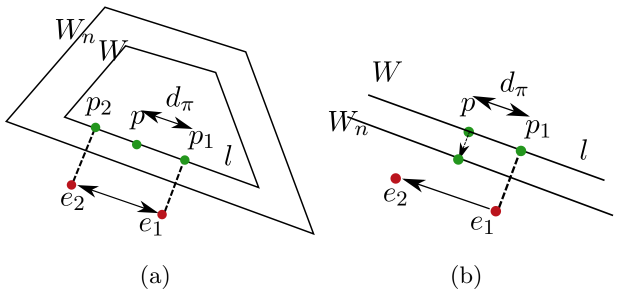


Figure 5.16: (a) The projection of e_1 and e_2 onto W are p_1 and p_2 respectively. (b) Progress in rook strategy.

Therefore, if we place the pursuer on the projection of the evader, then the evader cannot cross W without being captured (Lemma 5.3.14). Although locating p at $\pi(e, W)$ accomplishes our guarding goal, it makes it difficult to achieve the progress goal as follows. Suppose that the evader is in an edge region and let l be the corresponding edge in W. See Fig. 5.16(a). The evader can make it impossible for the pursuer to make progress by using the following strategy. The evader moves back and forth between e_1 and e_2 such that $e_1^i e_2^i$ is parallel to l. In response, the pursuer has to move between p_1 and p_2 (the projection of e_1 and e_2 respectively) to stick to its strategy of staying on the projection of the evader. Since the segment $e_1^i e_2^i$ is parallel to l and also because the length of $e_1^i e_2^i$ can be one, the length of $p_1 p_2$ is one. Consequently, the pursuer has to use all of its one unit of motion for guarding W: Nothing is left for it to make progress and move to W_n. The evader can repeat this for infinitely many steps, and thus it can escape forever against the pursuit strategy of staying on the projection.

We resolve the aforementioned problem by placing the pursuer "close" to $\pi(e, W)$ instead of "exactly" at $\pi(e, W)$. In particular, we place the pursuer on W at distance $d_\pi > 0$ from the projection of the evader onto W (to the left side of $\pi(e, W)$). For example in Fig. 5.16(a), if the evader is at e_1 the pursuer is located at p instead of p_1. Under certain conditions on d_π, the pursuer can still prevent the evader from crossing W. The pursuer can accomplish even more by making progress to the next wavefront in certain events such as when the evader moves to the left (Fig. 5.16(b)). We refer to this idea as the *rook strategy*. We present the details of guard and progress components of the strategy in Section 5.3.4 and Section 5.3.5 respectively. Our main effort in this paper is dedicated to providing the necessary conditions on d_π.

Finally, our pursuit strategy has another design parameter in addition to d_π: the discretization distance D which is the distance between two consecutive wavefronts (we will define the distance between two wavefronts in Section 5.3.3). We show that d_π and D must satisfy constraints that are functions of the terrain geometry (Lemma 5.3.5, Section 5.3.5.1, Lemma 5.3.8 and Lemma 5.3.10).

5.3.3 Discretization of the Surface into Wavefronts

We now study the discretization of $\mathcal{S}$ onto wavefronts. To do so, we move a plane which is parallel to the XY-plane downwards along the z direction starting from the highest point of the terrain. Notice that this plane is moved continuously. We then look at the images of the corresponding wavefronts in the XY-plane.

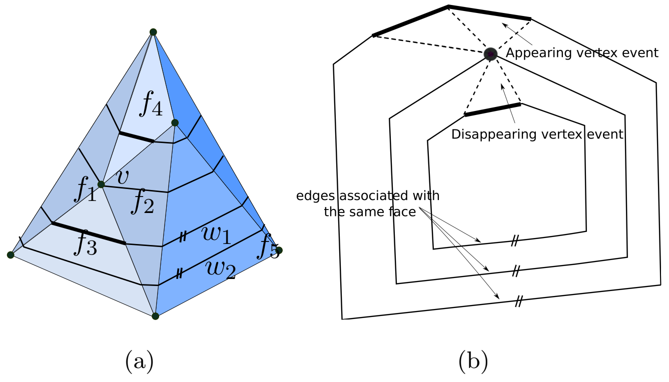


Figure 5.17: (a) A vertex event at v: The edge associated with face f_4 disappears. Then a new edge associated with f_3 appears. (b) The image of the wavefronts is shown.

We show that the changes in the combinatorial structure of these images occur at the vertices of $\mathcal{S}$. We refer to these changes as the discrete events. The set of wavefronts is then determined with regard to these discrete events.

In particular, before encountering a vertex two consecutive wavefronts are polygons that are *similar* to each other. Here, by *similar* we mean the following. Let W_1 and W_2 be two wavefronts such that there is no terrain vertex in between them (Fig. 5.17(a)). Then W_2^i is obtained from W_1^i by shifting all the edges of W_1^i in parallel. The shifting amount can be different for each edge (Fig. 5.17). When the frontier reaches a vertex in $\mathcal{S}$, we still have this parallel shifting pattern. However, some edges disappear from the frontier, and then new edges appear as the frontier passes the vertex (Fig. 5.17). We now formalize these events as follows.

Consider the wavefront W at height z and let $F(z)$ denote the set of faces that intersect W. We use $F(W)$ to denote the same set of edges if W is clear from the context. Also, let w be an edge in W which lies on the face $f \in F(z)$ (Fig. 5.17(a)). We refer to w as the edge in W which is *associated with* the face f. Now, as W is moved downwards, there can be no vertex on its way (*no vertex* event), or it will encounter a vertex (*vertex* events).

No vertex event: Let W_1 and W_2 be two wavefronts at heights

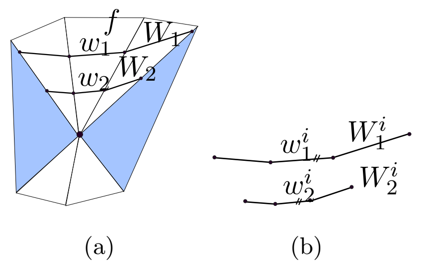


Figure 5.18: (a) W_1 and W_2 on $\mathcal{S}$. (b) Their images.

$z+\epsilon$ ($\epsilon > 0$) and z respectively such that $F(z) = F(z+\epsilon) = F$ (Fig. 5.18(a)). In other words, there is no vertex in $\mathcal{S}$ at height between z and $z+\epsilon$. Let f be a face in F, and let w_1 and w_2 be the two edges in W_1 and W_2 respectively that are associated with f. Then, the images of w_1 and w_2 in the XY-plane are parallel to each other (Fig. 5.18(b)). Moreover, this observation is true for all edges of W_1 and W_2.

Disappearing and appearing vertex events: Let v be a vertex of $\mathcal{S}$ which is at height z, and let W be the wavefront at z. Notice that if there are multiple vertices at the same height, we have multiple vertex events at the same time, one for each vertex. The argument below is still valid in this case.

Let W_u be the wavefront at $z+\epsilon_1$ ($\epsilon_1 > 0$) such that for heights h in $z < h \leq z+\epsilon_1$ we have $F(z+\epsilon_1) = F(h) = U$ (Fig. 5.19(a)). Next, denote the two faces that are adjacent to v in W by f_1 and f_2 (Fig. 5.19(a)). Let U' be the subset of U which is adjacent to v excluding f_1 and f_2 ($U' \subset U - \{f_1, f_2\}$). Then, as the frontier wavefront moves from $z+\epsilon_1$ to z, the edges in W_u^i that are associated with U' disappear in W^i. See Fig. 5.19(b). We refer to this event as the *disappearing vertex event.*

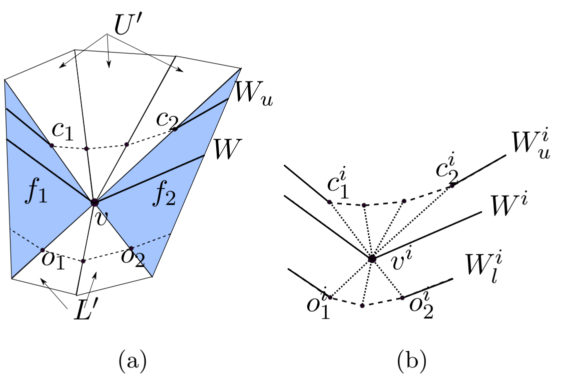


Figure 5.19: (a) the surface $\mathcal{S}$. (b) the XY-plane.

Similarly, let W_l be the wavefront at $z - \epsilon_2$ ($\epsilon_2 > 0$) such that for heights h in $z - \epsilon_2 \leq h < z$ we have $F(z - \epsilon_2) = F(h) = L$. Let $L' \subset L - \{f_1, f_2\}$ be the set of faces that are adjacent to v excluding f_1 and f_2 (Fig. 5.19). Then, as the frontier moves from z to $z - \epsilon_2$ new edges associated with L' appear in W_l^i. We refer to this event as the *appearing vertex event.*

The following Definition will be useful later when we define the distance between two wavefronts.

Definition 14 (The closing and opening wavefront vertices)**.** *Consider the portion of W_u that intersects U' (Fig. 5.19). We denote the two wavefront vertices on W_u that enclose this portion by c_1 and c_2. We also refer to them as the* closing *vertices . Similarly, o_1 and o_2, the* opening *vertices, are defined on W_l.*

Now that we have the discrete events, we are ready to define the distance between two consecutive wavefronts.

Distance Between Two Wavefronts: The distance between two consecutive wavefronts W and W_n, which is denoted by $d(W, W_n)$, is defined as follows for each of the three types of transitions between W and W_n:

No vertex event: Let us denote the image of the wavefront vertices in W by $\{w_1, w_2, \cdots, w_k\}$. Similarly, denote the vertices in W_n by $\{w'_1, w'_2, \cdots, w'_k\}$ (Fig. 5.20(a)). Then, $d(W, W_n)$ is defined as $\frac{1}{\alpha} \max_{1 \leq j \leq k} w_j w'_j$.

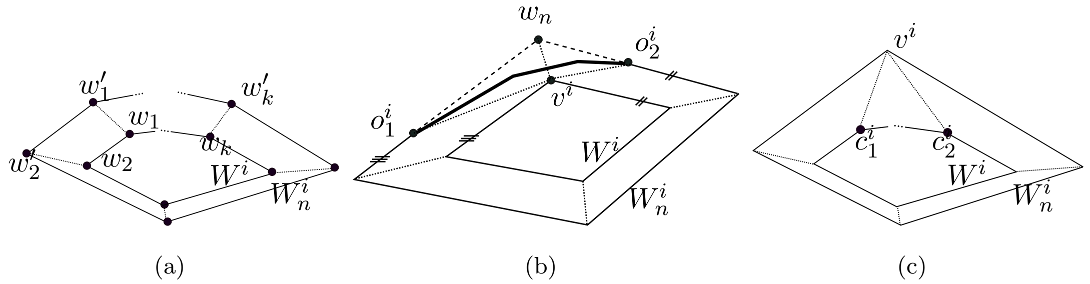


Figure 5.20: Images of W and W_n are shown in the base plane. (a) No vertex event. (b) Appearing vertex event. The appearing edges on W_n^i are shown in thicker line. (c) Disappearing vertex event.

Appearing vertex event: See Fig. 5.20(b). We define an auxiliary polygon in the XY-plane based on the image of W_n. Let us denote this new polygon by W_a. Then W_a is obtained by taking the extension of the two edges in W_n that are adjacent to o_1 and o_2. Let v_n be the intersection of these two extension lines. Then, $d(W, W_n)$ is defined as the maximum of $d(W, W_a)$ (from the previous no vertex event definition) and $\frac{1}{\alpha}\max(v^i o_1^i, v^i o_2^i)$.

Disappearing vertex event: See Fig. 5.20(c). Similar to the no vertex event case, let w_i and w_i' denote the rest of the wavefront vertices on W and W_n respectively. Then, $d(W, W_n)$ is defined as the maximum of $\frac{1}{\alpha}\max(v^i c_1^i, v^i c_2^i)$ and $\frac{1}{\alpha}\max_{1\leq j\leq k} w_j w_j'$.

We are now ready to present the discretization of $\mathcal{S}$ by wavefronts.

Discretization of $\mathcal{S}$ by the Wavefronts: In order to discretize $\mathcal{S}$, we need the discretization distance D as the input parameter. For a given value of D, we can choose the wavefronts on $\mathcal{S}$ such that the distance between any two consecutive wavefronts is at most D.

Specifically, the procedure for obtaining the wavefronts is the following. We add the highest point as the first wavefront. Starting from this first wavefront, we move downwards until the wavefront distance is D or we reach a terrain vertex. We then add the wavefront at this height as the second wavefront. We continue with this procedure until we reach the XY-plane and the surface of $\mathcal{S}$ is completely swept.

In our strategy we use the following property:

Lemma 5.3.4 (Distance between projections onto two consecutive wavefronts)**.** *Let W*

and W_n be two consecutive wavefronts and e be a point outside W_n. The distance between p_1, the image of the projection of e onto W, and p_2, the image of the projection of e onto W_n, is less than D where D is the maximum distance between any two wavefronts in the XY-plane.

Proof. The point e can be in an edge region or a wedge region of the wavefront W. In each case, we show that the distance between p_1 and p_2 is less than D (Lemma 5.3.11 and Lemma 5.3.12). □

5.3.4 Guarding Wavefronts

In this section, we present the pursuer's strategy for guarding the current wavefront W. The pursuer locates itself on W at distance $d_\pi > 0$ along W away from the projection of the evader onto W. In order to prevent the evader from crossing W, the distance d_π must satisfy a constraint that we present in this section. We call this configuration for guarding W the *rook* configuration which we formalize as follows. Suppose that the evader is outside the region enclosed by W (i.e. the evader is below W). Also, suppose that the pursuer is on the wavefront W. Consider the projection of e onto W which is denoted by $\pi(e, W)$.

Definition 15 (The Rook Configuration on $\mathcal{S}$)**.** *The pursuer on the wavefront W is in rook configuration if $d_\pi = d_W(p, \pi(e, W)) \leq \frac{\alpha}{2}$ where α is the minimum length coefficient. Fig. 5.21(a) depicts an illustration (See Definition 9 and Proposition 5.3.2 for d_W and α respectively.).*

In the following, we first show that when the pursuer is in the rook configuration, it can capture the evader if it tries to cross the wavefront W (Lemma 5.3.5). We then show that once the pursuer establishes the rook configuration on the frontier W, it can maintain it afterwards as the evader moves (Lemma 5.3.6).

Lemma 5.3.5. *Suppose that the pursuer is on the wavefront W in rook configuration (i.e. at distance $d_\pi \leq \frac{\alpha}{2}$ along W away from the projection of e). Then, the evader cannot cross W without being captured.*

Proof. Our proof has two parts. First, we show that the condition $d_\pi \leq \frac{\alpha}{2}$ implies that either the evader is already captured or d_π is less than the shortest distance between the evader and the wavefront W. We use this result in the second part of the proof and show that the pursuer can capture e if it tries to cross W.

First part: We would like to show that if $d_\pi \leq \frac{\alpha}{2}$, then either we have capture or the distance between the evader and W is at least d_π. We first introduce a lower bound on the distance between e and W. We then show that this lower bound is more than d_π.

To find the lower bound, we take another step to find another lower bound in the base plane. We connect this lower bound in the base plane to the lower bound on $\mathcal{S}$ by means of Lemma 5.3.3 and Proposition 5.3.2. Consider the image of e and W in the base plane (Fig. 5.21(a)). In the base plane, we know that the image of the projection of the evader onto W (i.e. $\pi^i(e, W)$) is the closest point on W^i to the image of e (Lemma 5.3.14). Notice that the shortest path between e^i and $\pi^i(e, W)$ is a line segment (since $\mathcal{S}$ is convex). Let us denote the length of this shortest path by h. Thus, all paths in the base plane are longer than h, and the length of the pre-image of each path in the base-plane is more than the length of its image. Therefore, all paths on $\mathcal{S}$ between p and e are longer than h.

We now show that $h > d_\pi$. To do so, we consider the triangle between the image of e, the image of p and the image of the projection of the evader onto W in the base plane, and we use the triangle inequality as follows. See Fig. 5.21(b). Let a denote the length of the segment between e^i and p^i in the base plane and $a_\mathcal{S}$ denote the length of its pre-image on $\mathcal{S}$. Similarly, let m denote the length of the segment between p^i and the image of the projection of the evader. If $a_\mathcal{S} \leq 1$, the evader is already captured. Therefore, suppose that $a_\mathcal{S} > 1$. Thus, $a > \alpha$ (Lemma 5.3.3). According to triangle inequality we have $m + h \geq a$. We also have $d_\pi \geq m$. Therefore $d_\pi + h \geq m + h \geq a$. Together with $a > \alpha$ we have $d_\pi + h > \alpha$. Now if $\frac{\alpha}{2} \geq d_\pi$, we conclude that $h > \alpha - d_\pi \geq \frac{\alpha}{2}$. Thus $h > \frac{\alpha}{2}$ and hence $h > d_\pi$. To recap, the shortest distance between the evader and the wavefront is at least h and h is more than d_π. Thus, d_π is a lower bound for the shortest distance between p and e.

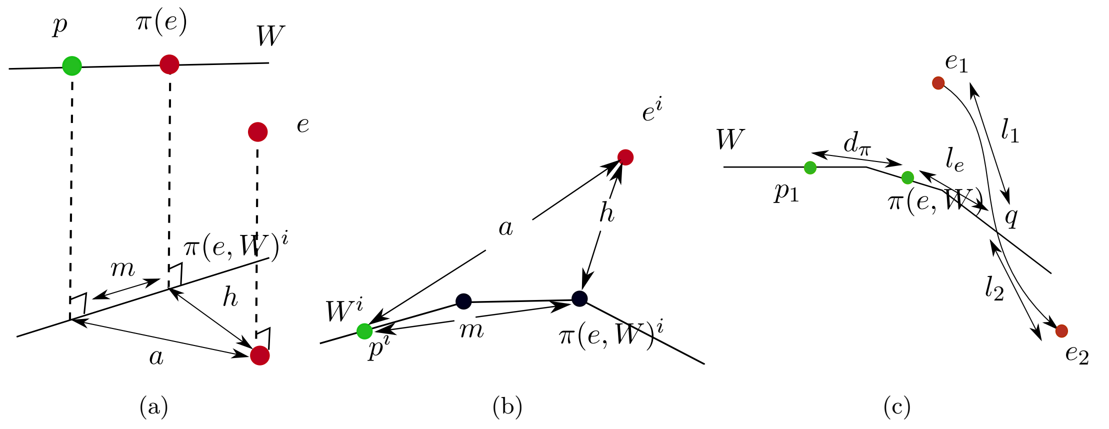


Figure 5.21: Proof of Lemma 5.3.5. (a) The players are shown on $\mathcal{S}$ along with their images on the base plane. (b) The images in the base plane. (c) If $d_\pi < h$, the evader cannot cross W without capture.

Second part: We next show that if the evader crosses W it will be captured by the pursuer. We show that there exists a path on $\mathcal{S}$ from the pursuer to the evader which is shorter than two. Therefore, the pursuer can capture the evader by moving along this path for one unit (at the end of this move the distance between the players is less than step size and hence we have capture). Suppose that the evader moves from e_1 to e_2 and crosses W at point q. Let us denote the length of the evader path from e_1 to q by l_1. Similarly, let l_2 be the length of the evader path from q to e_2. Thus $l_1 + l_2 \leq 1$. See Fig. 5.21(c). We show that the length of the path composed of $W(p, q)$ and the evader path from q to e_2 is less than 2. Let us denote the length of this path by l_p. Notice that $l_p = d_\pi + l_e + l_2$ where l_e is the distance between the projection of the evader onto W and q along W (Fig. 5.21(c)). From the first part of the proof we know that $d_\pi \leq l_1$. Also, according to Lemma 5.3.14 we have $l_e \leq l_1$. Therefore, $l_p = d_\pi + l_e + l_2 \leq l_1 + l_1 + l_2 \leq 1 + 1$. Thus, at the end of the pursuer's turn the distance between the players is less than the step size. □

Finally, we show that the pursuer can keep up staying close to the projection of the evader onto W as the evader moves.

Lemma 5.3.6. *The distance that the projection of the evader onto W travels is less than the distance that the evader travels.*

Proof. Consider the partitioning the region outside W^i into the wedge regions and the edge regions (Fig. 5.15(b)). The distance that the evader travels in each region is more than the distance that its projection onto W travels. □

5.3.5 Making Progress

We now focus on the pursuer's strategy for making progress towards the next wavefront W_n which is based on the motion of the evader.

Remark 2. *Without loss of generality, we assume that the pursuer establishes the rook configuration by locating itself to the left of* $\pi(e, W)$. *We then use the clockwise direction, as the base direction for classifying the evader's motion.*

The evader's motion can be categorized into three types of events: 1) the evader does not move in its current turn, 2) the evader moves in the counter clock-wise direction in its current turn, 3) the evader moves in the clock-wise direction for the next $O(N)$ steps where N is finite and we will specify it later in Section 5.3.5.3. In all of these events, we show that the pursuer can move to the projection of the evader onto the next wavefront W_n ($\pi(e, W_n)$). After moving to $\pi(e, W_n)$, the pursuer needs only one extra step to locate itself in the rook configuration i.e. at distance d_π away from $\pi(e, W_n)$. The pursuer repetitively applies this strategy to make progress to the next wavefront. Therefore, the evader will be captured in finite number of steps.

The key property that we incorporate in order to show that the pursuer can move to the projection of the evader onto the next wavefront is that for all points e the distance between the projection of e onto two consecutive wavefronts is less than the distance between the wavefronts themselves (Lemma 5.3.4).

The result of our paper is the following theorem.

Theorem 5.3.7 (Capture on Convex Terrains). *Our proposed pursuit strategy guarantees capture in finite number of steps. Specifically, the pursuer captures the evader in* $O((\frac{D_\mathcal{S}}{D} + n). \frac{|\mathcal{S}|}{1 - d_\pi - D})$ *where* n *is the number of vertices on* $\mathcal{S}$, $|\mathcal{S}|$ *denotes the perimeter of* $\mathcal{S}$, $D_\mathcal{S}$ *is the diagonal of* $\mathcal{S}$, d_π *is the rook configuration distance, and* D *is the distance between wavefronts.*

Proof. In Lemma 5.3.9 we show that $N = \frac{|\mathcal{S}|}{1 - d_\pi - D}$. Therefore, after at most $N = \frac{|\mathcal{S}|}{1 - d_\pi - D}$ steps, the pursuer can move from W to W_n and update W to W_n. The distance between two consecutive wavefronts is D or less than D when we have a vertex in between them (according to the construction phase). Thus, we have at most $\frac{D_{\mathcal{S}}}{D} + n$ wavefronts. Hence, the capture time is $O((\frac{D_{\mathcal{S}}}{D} + n).\frac{|\mathcal{S}|}{1-d_\pi-D})$. □

5.3.5.1 Evader Stays Still

In this section, we consider the case that the evader remains still in its turn. The pursuer uses this extra step in order to make progress towards the next wavefront. The key idea is that the distance between the projection of e onto two consecutive wavefronts W and W_n is at most D (Lemma 5.3.4). Therefore, the distance between the pursuer and the projection of the evader onto W_n is at most $d_\pi + D$. This is because the pursuer is guarding W in the rook configuration and thus away from the projection of e onto W for d_π. Consequently, d_π and D can be chosen small enough such that the pursuer can move to the projection of e onto the next wavefront in one step.

As a result, if we design d_π and D such that $d_\pi + D \leq 1$ the pursuer can move to the new projection of the evader in only one step.

5.3.5.2 Evader Moves Counter Clock-wise

We now consider the case that the evader is moving in counter clock-wise direction in the current time-step. Similar to Section 5.3.5.1 we use the observation that the distance between the projection of any point onto two consecutive wavefronts is at most D (Lemma 5.3.4). Therefore, the pursuer can move to $\pi(e_n, W)$ along W and then from there it can go to $\pi(e_n, W_n)$ in only one step (Fig. 5.22) if d_π and D are designed properly.

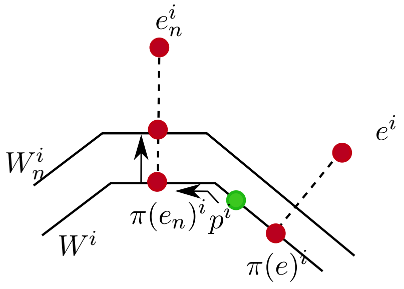


Figure 5.22: The evader moves counter clockwise.

Lemma 5.3.8. *Suppose that the evader moves counter clock-wise in its turn. Then, if we design d_π and D such that $\max(d_\pi, 1 - d_\pi) + D \leq 1$, the pursuer can move to $\pi(e_n, W_n)$ in one step.*

Proof. Consider the projection of e_n onto W: it can be to the left or to the right of p (Fig. 5.22). Therefore, the distance between p and $\pi(e_n, W)$ is at most $\max(d_\pi, 1-d_\pi)$. The distance between $\pi(e_n, W)$ and $\pi(e_n, W_n)$ is at most D (Lemma 5.3.4). Thus, d_π and D must satisfy: $\max(d_\pi, 1 - d_\pi) + D \leq 1$. □

5.3.5.3 Evader Moves in Clock-wise Direction for $O(N)$ Steps

We now consider the case that the evader is moving in the clock-wise direction for the next $O(N)$ where N is chosen such that after N steps it is guaranteed that either: (1) we have a time step such that the evader has to stay still or move counter-clockwise, or (2) the projection of the evader onto the current wavefront W circumnavigates around W for a complete round. In other words, if e moves clock-wise for N steps, $\pi(e, W)$ will come back to the same point on W.

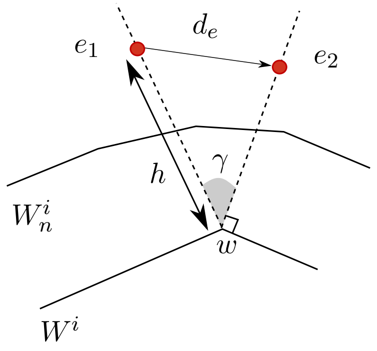


Figure 5.23: The pursuer can make progress as the evader crosses a wedge region.

Notice that if in one of these $O(N)$ steps, the evader stays still or moves counter clock-wise, then the pursuer can use the strategy in Section 5.3.5.1 and Section 5.3.5.2 for making progress towards W_n. Therefore, we can assume that during the next $O(N)$ the evader is moving clockwise. In this case, we show that the pursuer can make progress as follows. Since $\pi(e, W)$ circumnavigates around W, the evader crosses the wedge region of a wavefront vertex $w \in W$ (Fig. 5.23). We show that by properly selecting d_π and

D the pursuer can move to the projection of the evader onto the next wavefront W_n in one step. Intuitively, as the evader crosses the wedge region of w its projection onto W remains fixed i.e. w. Therefore, the pursuer does not need to move along W to maintain the rook configuration (which is staying close to the projection of the evader onto W). Instead, the pursuer moves toward the next wavefront W_n by moving to $\pi(e, W_n)$. In the following, we first compute N and then we present the condition on d_π and D for making progress.

We now compute the number of steps N that are required for moving clock-wise such that the projection of e onto W circumnavigates around W for a complete round. In other words, if e moves clock-wise for N steps, $\pi(e, W)$ will come back to the same point on W.

Lemma 5.3.9. *Consider the next $O(N)$ steps where $N = \frac{|\mathcal{S}|}{1-d_\pi-D})$ and $|\mathcal{S}|$ denotes the perimeter of the boundary of $\mathcal{S}$ (Assume that $d_\pi + D < 1$.). Then, we will have at least one of the following events in these $O(\frac{|\mathcal{S}|}{1-d_\pi-D})$ steps: 1) for at least one turn e does not move, or 2) it moves counter clock-wise, or 3) e circumnavigates around W i.e. $\pi(e, W)$ comes back to the same point on W.*

Proof. Suppose that we don't have none of the the first and the second events. We show that for sure we will have the third event. Since the first and the second events did not occur, the evader is moving clock-wise. Let e and e_n denote the location of the evader before and after a turn. For simplicity let us use the notations $d_e = d_W(\pi(e, W), \pi(e_n, W))$ and $d_\pi = d_W(p, \pi(e, W))$. Consider the path $\mathcal{S}(p, \pi(e_n, W_n)) = W(p, \pi(e, W)) + W(\pi(e, W), \pi(e_n, W)) + \mathcal{S}(\pi(e_n, W), \pi(e_n, W_n))$. The length of this path is $d_\pi + d_e + D$. Let us denote this length by d_p. There will be two case: 1) $d_p \leq 1$, 2) $d_p > 1$. In the first case, if $d_p \leq 1$, the pursuer can move to $\pi(e_n, W_n)$ and make progress to the next wavefront W_n.

In the second case we have $1 < d_p = d_\pi + d_e + D$. Consequently, $1 - d_\pi - D < d_e$. In this case, the evader's projection onto W moves for at least $1 - d_\pi - D$. Notice that the perimeter of W is at most $|\mathcal{S}|$ since the boundary of $\mathcal{S}$ and W are convex polygons. Therefore, after at most $\frac{|\mathcal{S}|}{1-d_\pi-D}$ steps, $\pi(e, W)$ comes back to the same point on W. Notice that here the following condition is required: $0 < 1 - d_\pi - D \Rightarrow d_\pi + D < 1$. □

Next, we show that if $\pi(e, W)$ circumnavigates around W, the pursuer can make progress to the next wavefront W_n by moving to $\pi(e, W_n)$.

Lemma 5.3.10. *Suppose that the evader moves in the same direction (clockwise) such that its projection onto W comes back to the same point on W. Then, if d_π and D satisfy the following inequality the pursuer can move to the projection of the evader onto the next wavefront W_n: $d_\pi + D \leq (\alpha - d_\pi)\sin\gamma$ where γ is the wedge angle (Definition 11) in $\mathcal{S}$ with minimum $\sin\gamma$ and α is the minimum length coefficient (Proposition 5.3.2).*

Proof. Since the projection of the evader onto W circumnavigates for a complete round around W, the evader crosses the wedge region of a wavefront vertex $w \in W$. Let us denote the image of the evader in the base plane by e_1 and e_2 as it crosses the corresponding wedge region (Fig. 5.23). In the following, we first find a lower bound on the length of the evader path e_1e_2. We then show that there is path between the pursuer and the projection of e_2 onto W_n of length at most $d_\pi + D$. By choosing d_π and D such that $d_\pi + D$ is less than the lower bound on the length of the evader path e_1e_2 we ensure that the pursuer can move to $\pi(e_2, W_n)$ and thus make progress to W_n.

We now present the lower bound on the length of the evader path e_1e_2. Consider the segment in between the images of p and e_1 in the base plane. Since $\mathcal{S}$ is convex and also according to Proposition 5.3.1, the pre-image of this segment is a valid path on $\mathcal{S}$. Let us denote the length of this path on $\mathcal{S}$ from p to e_1 by $a_{\mathcal{S}}$. Also, let a be the length of the image of this path (the segment in the base plane). Since the evader is not captured, it must be that $1 < a_{\mathcal{S}}$. Therefore, $\alpha < a$ (Lemma 5.3.3). Next, let h be the length of the segment e_1w^i in the base plane, and d_e be the length of the evader path $\mathcal{S}(e_1, e_2)$. Notice that the pursuer is in the rook configuration. Thus, the distance $d_W(p, w) = d_\pi$. Therefore, using the triangle property, we have $d_\pi + h \geq a$. Thus, $h \geq \alpha - d_\pi$. Observe that the length of the evader path between e_1 and e_2 is at least $h \sin\gamma$ (See Fig. 5.23). Therefore, $d_e \geq h\sin\gamma \geq (\alpha - d_\pi)\sin\gamma$. Therefore, the path that the evader travels on $\mathcal{S}$ from e_1 to e_2 is longer than $(\alpha - d_\pi)\sin\gamma$.

Next, we present the pursuer path to $\pi(e_2, W_n)$. Observe that the projection of e_1 and also e_2 onto W is w. Since the pursuer is in the rook configuration on W the distance between the pursuer and w along W is d_π. Also, the distance between $\pi(e_2, W) = w$ and $\pi(e_2, W_n)$ is at most D (Lemma 5.3.4). Therefore, the pursuer path $W(p, w) + \mathcal{S}(w, \pi(e_2, W_n))$ is shorter than $d_\pi + D$. Consequently if we design d_π and D

such that $d_\pi + D \leq (\alpha - d_\pi)\sin\gamma$, then the pursuer path will be shorter than the evader path. Therefore, we must have: $d_\pi(1+\sin\gamma) + D \leq \alpha\sin\gamma \Rightarrow d_\pi < \frac{\alpha\sin\gamma}{1+\sin\gamma}$. Hence, the pursuer can move to $\pi(e_2, W_n)$ as a response to evader motion from e_1 to e_2. $\square$

We next provide the proofs of lemmas that we skipped in the section.

5.3.6 Correctness Proofs

We now present the proofs of our lemmas. In the following, we treat the disappearing vertex event as a no vertex event with edge length zero for the disappearing edges.

Lemma 5.3.11. *Let W_1 and W_2 be two consecutive wavefronts, and e be a point outside W^i in the XY-plane. Suppose that e is in the edge region of an edge m_1 in W_1. Then, the distance between p_1, the image of the projection of e onto W_1, and p_2, the image of the projection of e onto W_2, is less than D where D is the maximum distance between any two wavefronts in the XY-plane.*

Proof. Fig. 5.24 shows the images of the wavefronts as well as the required points in the XY-plane. Let m_2 be the edge in W_2 which is parallel to m_1 (This edge exists in both cases of the no-vertex event and the appearing vertex event. For the disappearing event). The segment ep_1 is perpendicular to m_1 (Definition 10). There are two cases with regard to the location of m_2: (1) m_2 intersects ep_1 , (2) m_2 does not intersect ep_1.

1. m_2 intersects ep_1: This case is illustrated in Fig. 5.24(a). In this case, p_2 is in fact the intersection point between m_2 and ep_1. Therefore, the length of the segment p_1p_2 is less than D.

2. m_2 does not intersect ep_1: In this case without loss of generality suppose that m_2 is to the right of ep_1. The argument for the other case, when m_2 is to the left of ep_1, is similar. An illustration is shown in Fig. 5.24(b). Let q_1 and q_2 be the left endpoints of m_1 and m_2 respectively. We show that p_2 must be inside the rectangle made by m_1, the extension of m_2, ep_1 and the line parallel to ep_1 which passes through q_2, denoted by l. This rectangular region is shaded in Fig. 5.24(b). Therefore, $p_1p_2 \leq p_1q_2$ since $\widehat{ep_1q_1}$ is a right angle. Together with the fact that $p_1q_2 \leq q_1q_2$ (since $\widehat{q_1p_1q_2} \geq \frac{\pi}{2}$), we conclude that $p_1p_2 \leq q_1q_2 \leq D$.

Let us now prove that p_2 is inside the aforementioned rectangle. We show that p_2 cannot be to the left of ep_1 and also it cannot be to the right of l.

Before proceeding, notice that e must be above the line that contains m_2. Otherwise, if e is below m_2, the distance between p_1 and e will be less than D. Thus, the pursuer at p_1 capture the evader at e becasue their distance is less than 1.

First, we show that p_2 cannot be to the left of ep_1. For this purpose, assume the contrary and suppose that p_2 is to the left of ep_1 as shown in Fig. 5.24(c). Notice that since W_2 is convex, p_2 has to be below the line that contains m_2. Therefore, p_2 is inside the third quadrant made by m_2 and ep_1. Thus, $\widehat{ep_2q_2} \leq \frac{\pi}{2}$. Observe that p_2 and q_2 are both on W_2 and W_2 is convex. Therefore, the segment p_2q_2 is inside W_2. Hence, the edge on W_2 that contains p_2 must be inside the wedge made by ep_2 and q_2p_2. Therefore, the angle made by this edge and ep_2 is less than $\frac{\pi}{2}$ since $\widehat{ep_2q_2} \leq \frac{\pi}{2}$. But this is a contradiction because the angle made by ep_2 and W_2 must be a reflex angle (Lemma 5.3.13).

Finally, we show that p_2 cannot be to the right of l. Similar to the previous argument, assume the contrary: suppose that p_2 is to the right of l. Notice that p_2 is inside the fourth quadrant made by l and m_2. Therefore, $\widehat{ep_2q_2} \leq \frac{\pi}{2}$. Likewise to the proof above, $\widehat{ep_2q_2} \leq \frac{\pi}{2}$ contradicts the fact that the angle between W_2 and ep_2 is a reflex vertex.

$\square$

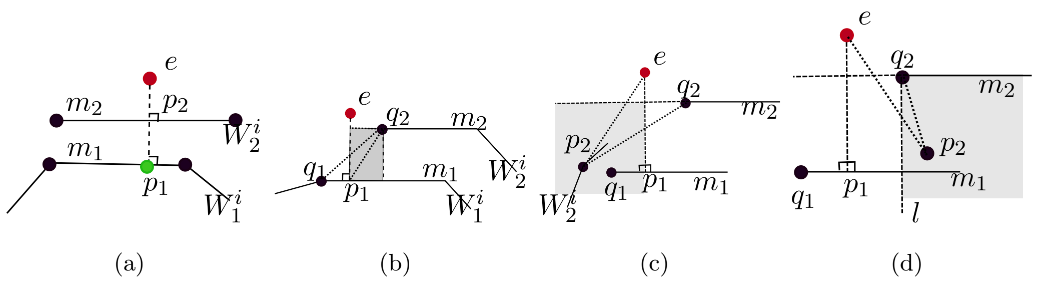


Figure 5.24: Proof of Lemma 5.3.11.

Lemma 5.3.12. *Let W_1 and W_2 be two consecutive wavefronts, and e be a point outside W^i in the XY-plane. Suppose that e is in the wedge region of a vertex w_1 in W_1. Then,* *the distance between $p_1 = w_1^i$, the image of the projection of e onto W_1, and p_2, the image of the projection of e onto W_2, is less than D where D is the maximum distance between any two wavefronts in the XY-plane.*

Proof. We prove the claim for the case that there is a no-vertex event between W_1 and W_2. The appearing and disappearing vertex events can be concluded from the no-vertex event. Therefore, suppose that the transition from W_1 to W_2 is a no-vertex event.

Let w_2 be the vertex in W_2 which corresponds to w_1 (w_2 is adjacent to edges in W_2 that are parallel to the edges in W_1 which are adjacent to w_1). Consider the circle centered at w_1 with radius $w_1w_2 \leq D$. We show that p_2 is inside this circle and thus the distance between $p_1 = w_1$ and p_2 is at most D. The main idea is the following. For contradiction, we assume that p_2 is outside this circle. We show that the angle $\widehat{w_2^ip_2e}$ is less than $\frac{\pi}{2}$. This contradicts the fact that the angle between W_2 and e_2p_2 must be at least $\frac{\pi}{2}$ (Lemma 5.3.13). We now present the proof that $\widehat{w_2^ip_2e} < \frac{\pi}{2}$. Let us denote the image of the two edges in W_1 that are adjacent to w_1 by m_1 and m_2. Also, let q_1 and q_2 be the image of the two edges in W_2 that are parallel to m_1 and m_2 respectively (and thus adjacent to w_2). We have two cases whether w_2 is inside the wedge region of w_1 or outside it:

1. w_2 is inside the wedge region of w_1: First, notice that $w_1^iw_2^i$ is inside the angle made by m_1 and m_2 as shown in Fig. 5.25(b). To see this assume the contrary as illustrated in Fig. 5.25(a) and observe that W_2^i cannot contain W_1^i which is a contradiction. Therefore $w_1^iw_2^i$ is inside the angle made by m_1 and m_2 (Fig. 5.25(b)). Let g_1 and g_2 be the intersection points between the circle and e_2p_2. Notice that $\widehat{e_2p_2w_2}$ is less than half of the arc length g_2w_2 (Fig. 5.25(b)). Notice that the arc g_2w_2 is smaller than π. Therefore, $\widehat{e_2p_2w_2} < \frac{\pi}{2}$.

2. w_2 is outside the wedge region of w_1: This case is illustrated in Fig. 5.25(c). Notice that the angle between m_1 and m_2 must be greater than $\frac{\pi}{2}$ because otherwise W_2^i cannot contain W_1^i. Similar to the previous case it can be shown that $\widehat{e_2p_2w_2}$ is less than $\frac{\pi}{2}$ which is a contradiction.

□

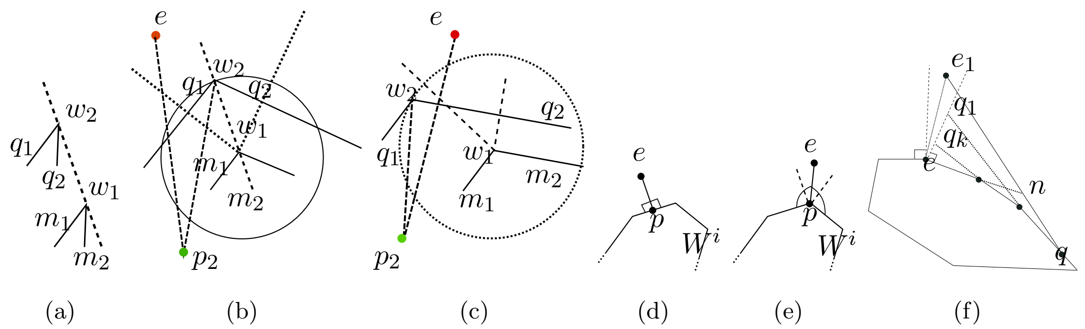


Figure 5.25: Proof of Lemma 5.3.12. **(a,b)** The case that w_2 is inside the wedge region of w_1. In (b) the arc g_2w_2 (the marked one) is smaller than π. **(c)** The case that w_2 is outside the wedge region of w_2. **(c,d)** Proof of Lemma 5.3.13. **(e)** Proof of Lemma 5.3.14.

Lemma 5.3.13. *Consider the image of a wavefront W in the XY-plane. Let e be a point in the XY-plane which is outside W^i. Let p be the projection of e onto W^i. The line segment ep makes two angles with W^i (Fig. 5.25(d) and Fig. 5.25(e)). Then both of these angles are larger than $\frac{\pi}{2}$.*

Proof. According to definition 10 there are two cases: the point e can be in an edge region or a wedge region. In both cases, the two angles made by ep and W^i are reflex angles. □

Lemma 5.3.14. *Consider the image of a wavefront W onto the XY-plane, i.e. W^i. Let e_1 be a point in the XY-plane which is outside the region enclosed by W^i. Moreover, let e denote the projection of e_1 onto W^i (i.e. $\pi(e_1, W)$, see Definition 12). Then, for all points $q \in W^i$, we have:*

- *In the XY-plane, e is closer to q than e_1. We show this by proving that $d_W(e, q) \leq d_{XY}(e_1, q)$.*
- *In the XY-plane, e is closer to e_1 than any other point q on W^i. In other words, $d_{XY}(e, e_1) \leq d_{XY}(q, e_1)$.*

Proof. First, notice that e_1 can be in an edge region or a wedge region (Definition 12). In both cases, the angle between e_1e and the edge(s) on W^i that contain e is at least $\frac{\pi}{2}$.

Next, consider the edges on W^i that are on $W^i(q, e)$ (Fig. 5.25(f)) and denote them by $\{m_1, \cdots m_k\}$. Consider the lines that contain these edges. Denote the intersection of these lines with the segment ee_1 by $\{q_1, \cdots q_k\}$. Observe that the angle between $m_j \in \{m_1, \cdots m_k\}$ and e_1e is greater than $\frac{\pi}{2}$. Therefore, it can be easily shown that $d_W(e, q) \leq d_{XY}(e_1, q)$.

Now, let us prove the second claim, that is $d_{XY}(e, e_1) \leq d_{XY}(q, e_1)$. Let n be the intersection of the edge that contains e and the segment e_1q. Since the angle between e_1e and en is at least $\frac{\pi}{2}$, it can be shown that $d_{XY}(e, e_1) \leq d_{XY}(n, e_1)$. Thus, $d_{XY}(e, e_1) \leq d_{XY}(q, e_1)$. $\square$

5.3.7 Summary

We showed that the class of convex terrains (height-maps with boundary) are single-pursuer-win when capture distance is non-zero. Our strategy is based on the rook strategy we discussed in Chapter 3. Intuitively, the pursuer starts from the highest point on the surface and guards wavefronts downward such that the surface is completely swept. The capture time of our proposed strategy is a function of the terrain's properties such as its height and maximum slope as well as the perimeter of its projection onto the base plane.

5.4 Concluding Remarks

In this chapter, we studied the lion and man game on the surface of a polyhedron. In Section 5.2 we studied general polyhedral surfaces with obstacles. We showed that if the capture distance is non-zero, three pursuers can capture the evader in finite time. Our strategy is based on a divide-and-conquer approach where two pursuers restrict the evader to a subset of the environment and the third pursuer divides the evader region to two smaller subsets. The evader is then confined to one of these smaller regions. The divide procedure continues until capture is achieved.

Our result is related to the result of Aigner and Fromme [28] who showed that in the game of cops and robbers, three cops suffice for capture on planar graphs. It might be tempting to use this result directly since the vertices and edges of a polyhedral surface with genus zero constitute a planar graph. It turns out that the conversion from the geometric version to the graph setup is not directly applicable for the following reasons. First, in the geometric version the players are not restricted to move only between the vertices. In fact, in the geometric version, the players can be anywhere in the continuous set of points on the surface and not just the discrete set of vertices. One might suggest to convert the geometric version to the graph setup by mapping the locations of the players to their nearest vertex on the surface. However, this mapping does not capture all possible movements of the players. For example, consider two points p_1 and p_2 on the surface that are at distance one from each other. These two points might be mapped to two vertices v_1 and v_2 that are not connected by any edge in the graph equivalent. As a result the movement between p_1 and p_2 cannot be described in the graph model.

In Section 5.3 we studied the lion and man game on convex terrains and presented a pursuit strategy which guarantees that the pursuer can reduce the distance between the players to the step size in finite time. One of the questions left open in this work is the optimality of this strategy. A second research direction is to characterize terrains in which a single pursuer suffices for capture. Even though convexity is sufficient, it is not necessary. A related question is to compute minimum number of pursuers for a given terrain.

Chapter 6

Stochastic Target (Probabilistic Search)

In Chapter 4 and Chapter 5 we studied variants of the lion and man game where the goal is to capture an adversarial target. In this chapter, we study the problem of finding a target which is simply moving randomly in a graph environment. That is, in each step it moves to one of its neighboring nodes according to a predefined probability distribution. We will investigate two classes of graphs: *linear graphs* (Section 6.2) and *two-dimensional grids* (Section 6.3). Our motivating application for this problem is our ongoing project on monitoring invasive carp in Minnesota lakes. Therefore, we also present our field experiments to demonstrate the applicability of our proposed model and search strategies for finding invasive carp.

Let us begin this chapter by giving a high-level description of the problem as well as our experimental system description.

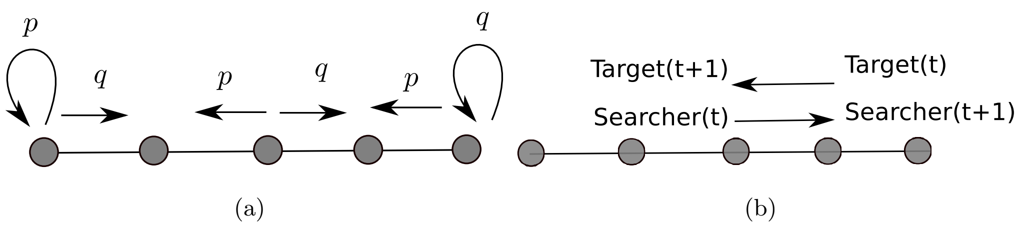


Figure 6.1: (a) Target's motion model is a simple random walk: with probability p it moves one step to the left, and with probability $q = 1 - p$ it moves one step to the right. (b) A crossing event is illustrated: at time t, the searcher is at node i, the target is at node $i+1$, and they move toward each other by taking the same edge in opposite directions.

6.1 Problem Statement and Experimental Setup

We first provide our search problem statement which is briefly to design a search strategy that maximizes the probability of finding a random walking target. Throughout the chapter, we will present more specification of the problem for linear graphs and grids. We then give an overview of our robotic system which is used in our field experiments.

6.1.1 Problem Statement

Let us begin by presenting the general formulation of the search problem. The searcher and the target are moving in a graph. We study two types of graphs: linear graphs in Section 6.2 and two-dimensional grids in Section 6.3. The searcher and the target move in turns, in discrete time steps and with equal speed. At each time-step, both of the players can move to one of their adjacent nodes or choose to stay in their current nodes. The target is doing a random walk i.e., the target moves to one of its adjacent neighbors according to a given probability. In the case of linear graphs (Section 6.2), we will study simple random walks where the target moves to the left or to the right with equal probability $\frac{1}{2}$. In the case of two-dimensional grids (Section 6.3) we investigate the case that the target is moving to its four adjacent neighbors or stays on its current node with equal probability $\frac{1}{5}$.

The direction that the searcher chooses to move is referred to as its *action*. In the case of linear graphs (Section 6.2), the set of actions we allow for the searcher are: `left`, `right, stay`. In the case of grids, we will consider *macro-actions* which are basically a sequence of actions (`left, right, up, down`), specifically a column to sweep (more details in Section 6.3).

We consider a maximum time T to complete the task. The objective of the searcher is to design a capture strategy that maximizes the probability of capture given the maximum search time T:

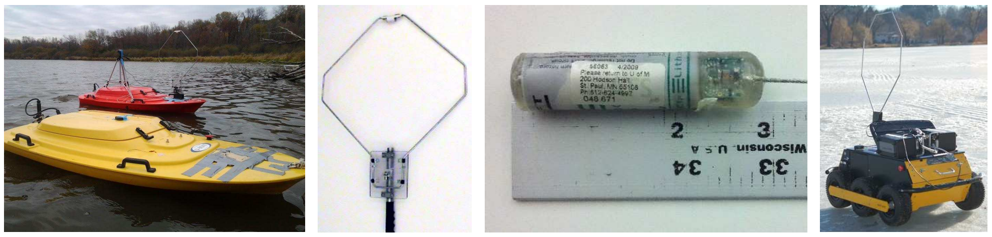
$$\max_{\Gamma} P_c(\Gamma = a_1 a_2 \cdots a_k) \quad s.t. \quad k \leq T\,, \tag{6.1}$$

where a_i denotes searcher's actions, $\Gamma = a_1 a_2 \cdots a_k$ is the searcher's strategy and $P_c(\Gamma)$ denotes the corresponding probability of capture.

6.1.2 Experimental Setup

In this section we present a description of our robotics platform including our sensor (directional antenna) and radio-tags used to detect the target. For our experimental demonstration, we will use Autonomous Surface Vehicles (ASV) in summer months and Autonomous Ground Vehicle (AGV) in winter months over frozen lakes. After presenting our proposed search strategies, we will provide our field experiments in Section 6.2.4 and Section 6.3.3.

Our first system, shown in Fig. 6.2(a), consists of two Autonomous Surface Vehicles (ASVs) carrying radio tracking equipment. The ASVs are built on boats manufactured by OceanScience [80] and were originally designed for remote operation. We added autonomous navigation with on-board GPS, digital compass and laptop, wireless communication via ad-hoc networking and remote override capabilities [81].



(a) (b) (c) (d)

Figure 6.2: (a) Two ASVs used in field experiments. The tracking equipment are installed on the boat that acts as the searcher. (b) The directional antenna used for sensing the target. (c) The tag used to track the target (in case of the fish the tag is implanted by surgery in the fish). The length of the tag is approximately 5 cm, and a 30cm antenna trailing off (From [1]) (d) The Husky A100 as our AGV.

For our experiments, one of the ASVs was designated as the searcher robot and the other used as the random-walking target. The searcher robot (the red boat in Fig. 6.2(a)) was equipped with a directional antenna (Fig. 6.2(c)), real time spectrum analyzer, and a laptop to process and track signals. The target robot (yellow in Fig. 6.2(a)) had a radio tag attached to a tow line. The radio tags transmit a low-power, uncoded pulse on a unique frequency approximately once per second. The antenna is directionally sensitive. To detect nearby tags, the antenna must be aligned with the tag. Our experiments are performed in Lake Staring, Minnesota.

As our Automatic Ground Vehicle (AGV), we used the Husky A100 built by Clearpath Robotics [82] which is a six wheel, two motor, differential drive machine. Fig. 6.2(d) shows the Husky equipped with the antenna.

We next study the search problem for finding a simple random walker in a linear graph.

6.2 Simple Random Walker on Linear Graphs: Crossing Detection Model

In this section, we study the random walk search problem in linear graphs[1]. The environment is a set of discrete locations $\{0, 1, \ldots, N\}$ equally distanced along a line segment. The target starts from an unknown node, and afterwards, it performs a *simple random walk* as follows: from location $0 < i < N$, with probability q it moves one unit to the *right*, and with the remaining probability $p = 1 - q$ it moves one unit to the *left*. Throughout this chapter, we mainly focus on the case of a symmetric random walk, i.e. $q = p = 0.5$, but most of our results can be easily extended to other values of p and q as well as to the case of a non-zero probability of staying at the current node (i.e. when $1 - q - p \neq 0$). In addition, the boundary points 0 and N are reflective (see Fig. 6.1(a)).

The searcher, on the other hand, starts from the left-most node $x(0) = 0$. At each time step, it can decide to move to the right, to the left or stay at its current node. Throughout the paper, we refer to these actions as R, L, S respectively. The searcher's strategy Γ is defined as a sequence of these actions $a \in \{R, L, S\}$. An example strategy can be $\Gamma = (R^i S^j L^l)^*$ which is move to the right for i steps, then stay for j steps, move to the left for l steps, and repeat forever.

The searcher is able to sense the target only on a node. As a result, crossing on edges, by which we mean taking the same edge in opposite directions, will not lead to capture (see Fig. 6.1(b)). We refer to this detection model as the *crossing model*[2]. In this model, traveling along the same edge as the target does not count as capture in this model. As a result, at any time the target can be on both sides of the searcher. This makes the problem of analyzing the capture probability challenging even when the search strategy is given.

In the rest of this section, we first present a method for computing the capture probability of a given strategy. The calculation is later used to compare the performance of the proposed search strategies. To find the optimal search strategy, we then formulate the problem as a POMDP. As it is well-known, the POMDP can be converted to a MDP by considering the searcher's belief, i.e. the probability distribution that the target is

[1] The material in this section appears in [6]

[2] Our results in the case of no-crossing detection model, i.e., when the target is detected also in crossing events, are presented in [6].

on each node, as the states. The resulting state space is large: it is exponential in the number of nodes. To deal with curse of dimensionality, we present two approximation methods. We refer to the first method as the belief-binning method. In this method, the intuition is that the shape of the belief at farther points is less important for the searcher in picking the best action at its current position. This is because by the time that the searcher reaches those points, the shape of the belief will change. In the second method, the problem is formulated as a Mixed Observability MDP (MOMDP) [58] where the fully observable variables of the state are separated from the partially observable ones. In both methods, the planning is done in a lower dimensional space. The two methods approach the problem differently and provide approximate solutions. However, both of them reveal a similar structure in their solutions (Section 6.2.3). We analyze the performance of the strategies in this particular structure in closed form (Theorem 6.2.1). We use this analysis to find the best set of parameters within the structure, and propose our final solution which is given in Table 6.3.

We next present the calculation method for computing the capture probability of a given strategy.

6.2.1 Capture Probability of a Given Strategy

To compare different search strategies given the crossing model, we first show how the probability of capture for a given strategy can be computed. Let us denote the searcher's location at time t by $s(t)$ and the target's location by $e(t)$. The target is initially at $e(0)$. The searcher is performing the strategy Γ given as a sequence of actions R, L and S. Fig. 6.3 shows an illustrative example of the location of the searcher for a specific strategy, and the target for a specific random walk path as a function of time.

We are interested in computing the probability of capturing the target at time t denoted by $P_c(t)$. The capture events are those that the searcher and the target are at the same position at the same time. Thus, we must consider the target's paths that end up at $s(t)$ at time t, i.e. $e(t) = s(t)$. Counting the events that the target starts at $e(0)$ and reaches $s(t)$ at t is not too difficult. However, we cannot simply sum up the probability of these events to obtain P_c because they include overlapping capture events with $e(k) = s(k)$ for multiple values of $k \leq t$. First, we must only count the events such that the target has not been captured sooner than t. Second, since crossing is allowed, path interactions such as the one marked by the arrow in Fig. 6.3 do not count as capture. In order to tackle these two challenges, we look at the searcher's path $s(t)$ as a piecewise constant function. That is, $s(t)$ is composed of a set of time intervals $[t_k, t_{k+1})$ such that $s(t)$ is constant in $[t_k, t_{k+1})$ and has the value s_k (the searcher is staying at s_k in $[t_k, t_{k+1})$). In Fig. 6.3 the searcher is at s_{k-1} in the time interval $[t_{k-1}, t_k)$.

In order to compute $P_c(t)$, we take a divide and conquer approach to recursively compute the target paths that yield capture at time t but are *safe* before t. By *safe* we mean that the target has not been captured before time t. To do so, we consider the intervals before t. Each time-interval acts as the basis in our divide and conquer method, and we can employ them to overcome the two issues mentioned above as follows. Since in each interval $s(t)$ is constant we do not have the crossing events such as the one marked by the arrow in Fig. 6.3. We also can enumerate the target paths that co-locate the target and the searcher for the *first* time at t by counting the target paths that are completely to the left or to the right of s_i during $[t_i, t_{i+1})$ for $t_i < t$. This enables us to compute the capture events using recursive equations as follows. Let us introduce the following probabilities:

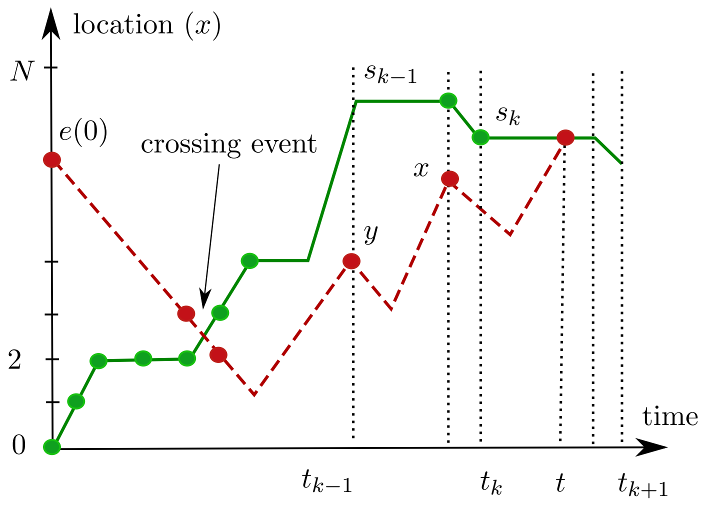


Figure 6.3: The position of the players as a function of time. The target's path is shown in dashed lines. The time marked by the arrow is not capture because crossing is allowed.

- $P_{safe}(x, t)$: the probability that the target *safely* arrives at location x at time t.

Here *safe* refers to the fact that the target has not been captured before time t.

- $P_{safe}(x_1, t_k, x_2, t_{k+1} - 1)$: This is the probability that the target *safely* reaches x_2 at time $t_{k+1} - 1$ assuming that it is at x_1 at time t_k. Notice that the target has to stay either to the left, or to the right of s_k during $[t_k, t_{k+1})$ in order to avoid capture and remain safe. Referring to Fig. 6.3, the target paths must be either below or above s_k at $[t_k, t_{k+1})$. Note that this function is different from the previous one in the sense that it counts the safe events in a single interval $[t_k, t_{k+1})$.

- $F(x_1, x_2, t)$: the probability that for the first time the target reaches at x_2 after t time steps starting from x_1 (first passage probability).

- $G(x_1, x_2, t)$: the probability that the target starts at location x_1 and reaches at x_2 after t time steps (not necessarily for the first time).

Our goal is to compute $P_c(t)$, the probability of capturing the target at time t. First, we consider the time interval $[t_k, t_{k+1})$ that t belongs to. See Fig. 6.3. Let x be the target's position at time $t_k - 1$. The searcher is at s_{k-1} at $[t_{k-1}, t_k)$. The capture events can be described as follows: The target safely arrives at x at time $t_k - 1$, i.e. $P_{safe}(x, t_k - 1)$ and then, from x it reaches s_k for the first time after $t - t_k + 1$ time steps:

$$P_c(t) = \sum_x P_{safe}(x, t_k - 1) F(x, s_k, t - t_k + 1)\,.$$

The safe events $P_{safe}(x, t_k - 1)$ can be obtained from the following recursive equations:

$$P_{safe}(x, t_k - 1) = \sum_y P_{safe}(y, t_{k-1}, x, t_k - 1) P_{safe}(y, t_{k-1})$$

The probability function $P_{safe}(x_1, t_{k-1}, x_2, t_k - 1)$ is computed as follows:

$$\begin{aligned} P_{safe}(x_1, t_{k-1}, x_2, t_k - 1) &= G(x_1, x_2, t_k - 1 - t_{k-1}) \\ - \sum_{t=0}^{t_k - 1 - t_{k-1}} &(F(x_1, s_{k-1}, t) G(s_{k-1}, x_2, t_k - 1 - t_{k-1} - t)) \end{aligned}$$

In other words, the events that the target crosses s_{k-1} (second term in r.h.s. of the equation above) are excluded from the total number of events (first term in r.h.s. of the equation above). Notice that the searcher is at s_{k-1} in the time interval $[t_{k-1}, t_k)$.

Finally, computing $G(x_1, x_2, t)$ and $F(x_1, x_2, t)$ is straightforward and we refer the interested reader to [61] for the corresponding equations.

6.2.2 Partially Observable Markov Decision Process

In this section we present the formulation of our search problem as a Markov Decision Process (MDP) [66]. Recall from Section 3.2.1 that an MDP is a tuple $(\mathbb{S}, \mathbb{A}, \mathbb{T}, \mathbb{R})$ where $\mathbb{S}$ is the set of possible states, $\mathbb{A}$ is the set of actions, $\mathbb{T}$ is the probability of transitioning between the states as a result of performing each action, and $\mathbb{R}$ is the reward collected for each transition.

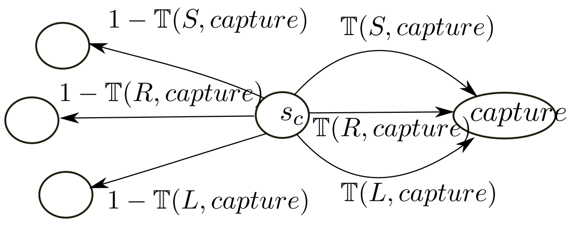


Figure 6.4: MDP state transitions. In the current state s_c, by performing $a \in \{R, L, S\}$, with probability $\mathbb{T}(a, capture)$ the searcher captures the target and with the remaining probability the next state will be s_n.

Formulating the search problem as an MDP, the states could be defined as (p, e) where p is the position of the searcher and e is the position of the target. However, we cannot use this setup since the location of the target is not observable to the searcher. The searcher's only observation is that it has not captured the target yet. Therefore, our problem is in fact a Partially Observable Markov Decision Processes (POMDP). However, we can convert it to an MDP by defining the states as the searcher's belief about the target's location [83]. The goal is to find the policy that maximizes the reward collected by the searcher upon execution of the policy. The following are the sets which define our MDP (Fig. 6.4):

- The states are defined as (B, E, s) where B is the belief of the searcher about the position of the target, E is the current energy of the searcher and s is the current position of the searcher. Here B is represented as the probability vector $B = [p_0, p_1, ..., p_N]$ where p_i is the probability that the target is at position i given that it is not captured yet. A set of terminal states is given by $\{s_{\text{capture}}, s_{\text{no-time}}\}$ where s_{capture} denotes the capture state, and $s_{\text{no-time}}$ denotes the state in which the robot runs out of time budget.

- The set of actions that the searcher can perform in each state are: *stay* at its current position, move one step to the *left*, or one step to the *right*.
- The transition probability matrix with entries $\mathbb{T}(s_i, s_j, a)$ that represent the probability that the searcher transitions to state s_j by performing action a in state s_i.
- The reward matrix which represents the transition reward from state s_i to state s_j after performing action a.

In the following, we describe the details of the state space and the proper definition of the reward function. We skip the calculation of the probability transition matrix since it is straightforward.

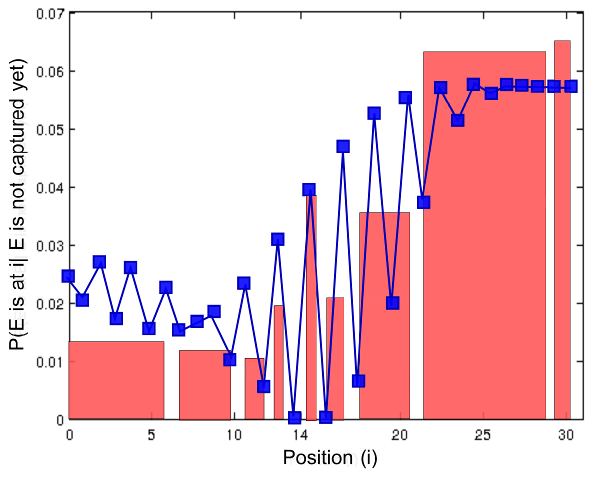


Figure 6.5: Blue is the actual belief and red is its approximation by bins. The searcher is at $x = 14$.

Method 1: The belief-binning approximation method: Initially, the searcher starts from the location $x = 0$. The searcher begins its mission with no information about the position of the target (except that the target is not captured yet). Therefore, the initial belief vector is the uniform probability distribution over $[0, N]$.

Note that the number of states will be exponential in the number of discrete levels used for representing the probability vector $B = [p_0, p_1, ..., p_N]$ (with the exponent equal to N, the size of the environment). To deal with this problem, one method is to approximate the belief by a specific function which can be represented by a small number of parameters. However, this approach cannot be applied to the crossing case directly, because here the belief is not a smooth function. A sample of the searcher's belief is shown in Fig. 6.5.

Intuitively, the value of the belief at the nodes that are far away from the searcher's current position is not effecting the best action. Based on this observation, we represent the belief by bins with exponentially increasing width as follows. There are two bins with width 2^i that start at nodes $s + 2^i$ and $s - 2^i$ respectively $(0 \leq i \leq \log(N))$. The approximate belief in each bin is uniform. To compute its value we first compute the cumulative belief in each bin and then we take the average of this cumulative value in the corresponding bin. Finally, we assign the closest discretization level to the average value as the bin value. An example is shown in Fig. 6.5.

The reward matrix: As expressed in (6.1), we are looking for the strategy that maximizes the probability of capture. In order to associate the value of MDP states with the probability of capture, the reward function is defined as follows. The transition reward from all states (except s_{capture} and $s_{\text{no-time}}$) to the capture state s_{capture} is one. All other transition rewards are zero. The state values of the aforementioned MDP is an approximation of the probability of capture. Therefore, the strategy that maximizes the state values, i.e. the solution to the MDP, is in fact the one that maximizes the probability of capturing the target.

The solution technique: Finally, we use finite-horizon MDP implementation available in INRA MDP MATLAB Toolbox [84] which uses backward induction algorithm. The number of stages is set to T, the time constraint. The terminal reward is set to zero for all states except s_{capture} which is set to one.

Method 2: The MOMDP approximation method: An alternative approach to tackle the large state space of our problem is to formulate the problem as a Mixed Observability MDP [58]. In this formulation, the state components that are fully observable are separated from the ones that are partially observable. As a result, the belief is maintained on a smaller set of variables, and the size of the state space can be reduced significantly.

Recall from Section 3.2.2 that a MOMDP is specified as a tuple $(\mathbb{X}, \mathbb{Y}, \mathbb{A}, \mathbb{O}, \mathbb{T}_x, \mathbb{T}_y, \mathbb{Z}, \mathbb{R})$ where $\mathbb{X}$ represents the set of fully observable components, $\mathbb{Y}$ represents the set of partially observable components, and $\mathbb{A}, \mathbb{O}$ are the set of actions and observations respectively. The function $\mathbb{Z}(o, s, a)$ is the conditional probability of observing o after performing action o and moving to the state s, $\mathbb{T}_x(x, y, a, x')$ represents the transition probability of the fully observable state component x, and $\mathbb{T}_y(x, y, a, y')$ is the transition probability of the partially observable component y.

Adopting the MOMDP formulation to our problem, the searcher's position and also the time budget are fully observable while the target's location is partially observable. We use the Approximate POMDP Planning (APPL) toolkit which is available at [73]. The APPL toolkit combines MOMDP with SARSOP which is a point-based POMDP algorithm [58].

6.2.3 Simulation Results

We are now ready to solve for the search strategies using the proposed formulations. We consider three cases: 1) T is enough for at least two sweeps of the line, i.e. $T > 2N$; 2) $T \leq 1.5N$; 3) $1.5N < T \leq 2N$. In the first case, we present intuitive strategies and show that they guarantee a high probability of capture. In the remaining cases, we provide the strategies obtained from the MDP formulations. The simulations for the MDP formulations are done on a Dell Poweredge 6950 machine with $14GB$ memory. In the belief-binning approach, we used 50 levels for discretizing each bin. Thus, if there are n bins, the number of possible states would be 50^n. In the MOMDP method, we let the solver run until a target precision[3] less than 0.001 is reached or a timeout happens after 3 hours of execution.

[3] Target precision is a function of $|\overline{V} - \underline{V}|$ for the set of samples in the SARSOP method.

When Time is Enough for Two Sweeps: In this case, the searcher has enough time for at least two sweeps $T > 2N$. We start by considering the case that $T = 2N+1$. One intuitive strategy is to sweep the whole line twice. However, if the searcher moves all the time, the parity of its distance to the target will never change, unless the target does a stay action at one of the boundary points (Fig. 6.1(a)). This is because, at other points, the target always moves one step to the right or to the left, and thus its distance to the searcher changes by two. Therefore, we add a wait step in order to increase the probability of capture in the event that the target starts from an odd node. We call this strategy the *Sweep* strategy: the searcher moves all the way to the right, waits for one step at the last point N and then moves back to the left toward the first node $(R^N SL^N ...)$.

We also analyze a second strategy, which we call *StopAndGo*: the searcher moves for one step, then waits for one step and so on $(RSRS...)$.

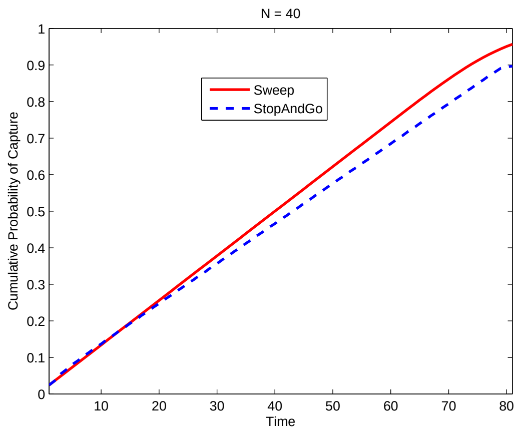


Figure 6.6: Cumulative probability of capture obtained from Section 6.2.1. Here $N = 40$.

The probability of capture for these two strategies is computed using the analysis in Section 6.2.1. Fig. 6.6 depicts the cumulative probability of capture for $N = 40$. As shown in this figure, these strategies achieve a very high probability of capture: 0.95 and 0.90 for Sweep and StopAndGo respectively. Since the highest possible performance for the probability of capture is one, we choose the Sweep strategy as our best strategy for this case. Notice that for larger values of $T > 2N$ we can repeat the proposed strategy until time exceeds T.

	$T = 9$	$T = 14$	$T = 19$	$T = 24$	$T = 29$
Belief-binning	$R\ S$ R^2SR^3S	$(RS)^2\ R^3S$ $(R^2S)^2$	$(R^3S)^4R^2S$	R^2SR^3S $(R^2S)^3(R^3S)^2$	$R^2SR^3S(R^2S)^3$ $(RS)^2R^3SR^2SRS$
P_c(Belief-binning)	0.2966	0.4201	0.5711	0.6991	0.8020
Uniform	$(R^3S)^2R$	$(R^3S)^3R^2$	$(R^5S)^3R$	$(R^3S)^6$	$(R^2S)^9S^2$
P_c(Uniform)	0.2971	0.4294	0.5642	0.7095	0.8031
MOMDP	$(R^2S)^2RSL$	R^2SR^3S $(R^2S)^2L$	$R^2S(R^3S)^3$ R^2SL	$R^2S(R^3S)^4R^2$ S^2L	$(R^2S)^6(R^3S)^2$ S^2L
P_c(MOMDP)	0.2179	0.3787	0.5323	0.6633	0.7873

Table 6.1: The MDP solutions for $N = 20$.

	$T = 14$	$T = 24$	$T = 35$	$T = 44$
Belief-binning	$(R^2S)^3R^4S$	$R^3SR^5SR^3$ SR^6SR^2S	$R^{30}S^5$	$RS^3R^{19}SR^3S$ $R^2S(RS)^2$
P_c(Belief-binning)	0.2830	0.4625	0.58	0.675
Uniform	$(R^3S)^3R^2$	$(R^3S)^6$	$(R^4S)^7$	$(R^2S)^{14}R^2$
P_c(Uniform)	0.2818	0.4656	0.6638	0.787
MOMDP	R^2SR^3S $(R^2S)^2L$	$R^2S(R^3S)^4$ R^2S^2L	$R^2S(R^3S)^2$ $(R^4S)^3R^5S^3L$	$RS(R^2S)^{10}R^6$ SR^2S^2L
P_c(MOMDP)	0.2421	0.4219	0.6333	0.7651

Table 6.2: The MDP solutions for $N = 31$.

When Time is less than 1.5 sweeps: In this case $T \leq 1.5N$. The strategies found by the two approximation methods for $N = 20$ and $N = 31$ are shown in Table 6.1 and Table 6.2 respectively.

A first interesting result is that, as shown in these tables, the solutions have a common property: the stay actions are uniformly distributed among the right actions. In other words, the strategies are of the form $(R^kS)^m$. Let us refer to this class of strategies as the uniform strategies. Table 6.1 and Table 6.2 also present the best uniform strategy for the same values of N and T which are obtained by changing the number of rights k in each group of (R^kS) and computing the probability of capture using the analysis in Section 6.2.1. Observe that the uniform strategy is very close to the solutions found by the two MDP methods.

To compare the belief-binning and the MOMDP performance, observe that for $N = 20$ in Table 6.1 the solutions of belief-binning method are all better than the MOMDP in terms of capture probability. This is true also in Table 6.2 for for $N = 31$ and $T = 14, 24$. However, for $T = 35, 44$ the MOMDP solution outperforms the belief-binning strategies which is due to the error corresponding to the resolution in binning levels. On the other hand, the MOMDP solutions exhibit the uniform structure in all instances which makes MOMDP a more suitable approach for larger values of N and T[4] .

Next, we focus on the problem of finding a good uniform strategy. In fact, we would like to find the correct number of *right* actions before each *stay* action and optimize k. In order to find the best k, we take the following approach. We first derive closed form equations to approximate the capture probability of $(R^kS)^m$. Our approximation is applicable for $k \geq 3$. Therefore, we use it to find the best $k \geq 3$. We then compare this best solution with $k \in \{1, 2\}$ both in simulations and also using our analysis (Section 6.2.1). From this comparison, we conclude our proposed optimal strategy of the form $(R^kS)^m$.

The following theorem, presents the aforementioned closed form. We present the details of the proof in [6].

Theorem 6.2.1. *The probability of capture for the strategy $(R^kS)^M$ where $T = M(k+1)$ and $3 \leq k$ can be approximated as:*

$$P_c \approx \begin{cases} \frac{M(k+2)}{2N} = \frac{T+M}{2N} & \text{if } Mk < N \\ \frac{T+M-1}{2N} & \text{if } Mk = N \end{cases} \tag{6.2}$$

Therefore, for a given T the capture probability is maximum when M takes its largest possible value. That is the optimum number of rights in each group of R^kS is $k = 3$ when $k \geq 3$.

Next, let us compare the performance of $k = 3$ with $k \in \{1, 2\}$. Fig. 6.7 shows this comparison for the following strategies: $(RS)^{35}$ and $N = 35$, $(R^2S)^{17}$ and $N = 34$,

[4] Note that the APPL toolkit can handle instances where $N < 100$ and $T < 40$.

$(R^3S)^{12}$. This comparison suggests that the performance of $(R^2S)^m$ is comparable to the other uniform strategies and sometimes it is the best. Thus, we pick $(R^2S)^m$ as our proposed strategy when $T < 1.5N$.

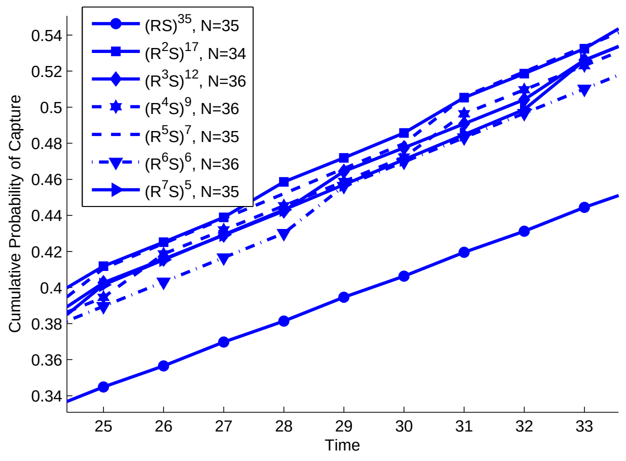


Figure 6.7: Comparison of uniform strategies $(R^kS)^m$. Here $N = km$ to prevent extra R at the end of the strategy.

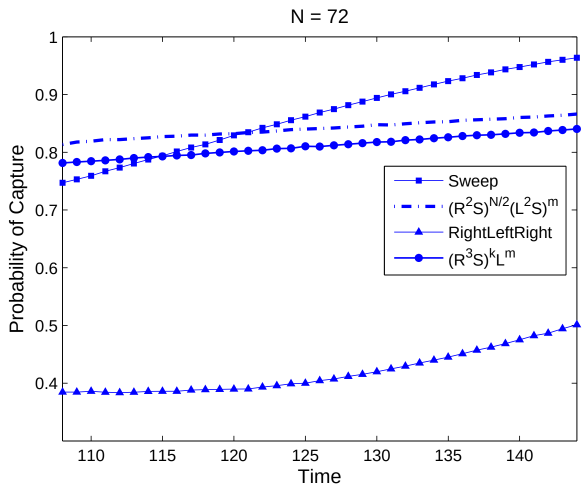


Figure 6.8: Comparison of four strategies when $1.5N < T$ for $N = 72$ obtained from simulation.

When Time is greater than 1.5 sweeps

Let us now investigate the case that $1.5N \leq T < 2N$. We compare four strategies.

The first one is the Sweep strategy we introduced earlier. Then, we define the following strategies:

- $(R^2S)^{\frac{N}{2}}(L^2S)^m$: repeat the pattern R^2S until the searcher is at node N, then repeat the pattern L^2S for the rest.
- $(R^3S)^kL^m$: repeat the pattern (R^3S) until the searcher is at node N, then move back to the left.
- `RightLeftRight`: move to the right for $\frac{3N}{4}$, then turn back and move to the left for $\frac{N}{2}$, and then move to the right for the rest of time-steps.

As shown in Fig. 6.8 for $N = 72$, the performance of $(R^2S)^{\frac{N}{2}}(L^2S)^m$ is better for $T < 120$, and then afterwards the Sweep strategy becomes better. It is worth emphasizing that both are above 0.81 after time 120. We observed the same behavior for $N = 30$ and $N = 54$. Thus, we declare both of them as our candidate strategies for the case that $1.7N < T$. When $1.5N < T < 1.7N$, we consider $(R^2S)^{\frac{N}{2}}(L^2S)^m$ our best strategy.

A summary of the best strategies found for different cases is depicted in Table 6.3.

	Best Strategy
$2N < T$	Sweep
$1.7N < T \leq 2N$	Sweep
$1.5N < T \leq 1.7N$	$(R^2S)^{\frac{N}{2}}(L^2S)^m$
$T \leq 1.5N$	$(R^2S)^m$

Table 6.3: Summary of the proposed best strategies for different cases.

6.2.4 Experimental Demonstration

We now present experiments to demonstrate the applicability of our proposed search strategies in the carp monitoring project. Here the target is the common carp which are tagged and can be detected by our directional antennas (See Section 6.1.2). We conduct our experiments in Minnesota lakes instead of rivers for the following reasons. In addition to safety reasons for our autonomous robots, we expect that considering a

1D search path is a reasonable assumption for this application because these fish tend to move near the shore (where there is more vegetation) most of the time [85]. See Fig. 6.9 for an illustration. We present preliminary results which are mainly focused on modeling. First, we provide evidence that taking measurements on a discrete set of nodes produces reliable detection. In particular, as the robot stops and turns off its motor, the noise interference which is a main reason for false positives is reduced. It must be noted that an important feature of our system is the directional sensitivity of the antenna. That is, our sensor must be aligned with the target for maximum signal strength received from the target. As a first approach, we chose to rotate the antenna and take measurements at multiple orientations when the robot is stopped. This simplifies the modeling as we no longer need to model the antenna orientation. We present results from the summer months using Autonomous Surface Vehicles (ASV) and winter months using an Autonomous Ground Vehicle (AGV). Finally, it must be noted that in the case of ASVs the ratio of movement cost to stationary keeping cost is $c_m/c_s = 5.7/0.2$ while in the case of AGVs the stationary keeping has zero cost.

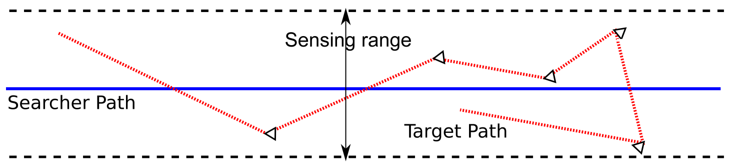


Figure 6.9: The target is performing a random walk on a corridor of width equal to the sensing range (red dotted path). The searcher moves on the corresponding line segment (blue path).

6.2.4.1 Experiments with the ASVs

In the following experiment, the searcher sweeps an L-shaped corridor of length $320m$ as shown in Fig. 6.10. For safety reasons, we use this short length in order to keep the ASVs within the communication range.

The searcher sweeps the path from the first node to the last node such that after uniform time intervals it turns off its motor and takes measurements by rotating the antenna. This will give us a comparison between the signal strength during motion versus waiting, and ultimately the effect of the noise interference from motor. In the following plots, we use the propeller speed as an indicator of the time intervals that the boat has turned off its motor.

Fourteen radio frequencies (corresponding to known tags which were not included in the trials) were monitored for transmissions, while the target transmitted only at frequency 49611 KHz. Notice that in a realistic search for a target, the target frequency is unknown. Therefore, we need a comparison between the true and a noise frequency that we know it is not present in the trial.

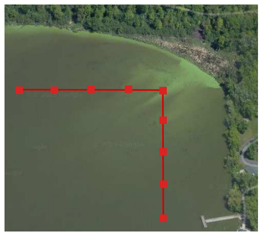


Figure 6.10: The experiment area which includes 9 nodes with distance $40m$ (total length is $320m$) along an L-shaped path.

To determine detection, we use the following method. For each frequency that the antenna is listening to, let m and σ be the corresponding mean and variance for the signal strength respectively. Also, let f be the current signal strength. Consider the following criterion for all frequencies:

$$\frac{f - m}{\sigma} \tag{6.3}$$

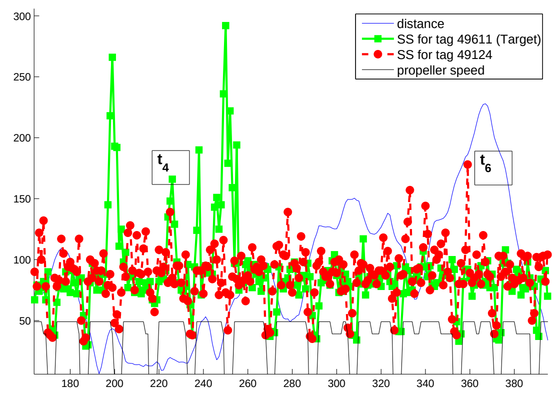


Figure 6.11: The Signal Strength (SS) for the target tag (49611 KHz) and 49124 KHz that is not present in the trial. Also, the distance between the searcher and the target are shown. Notice that we have false negative at $[t_2, t_3]$ although the target is close to the searcher (around $15m$). One reason for a false negative could be mis-alignment between the antenna and the tag.

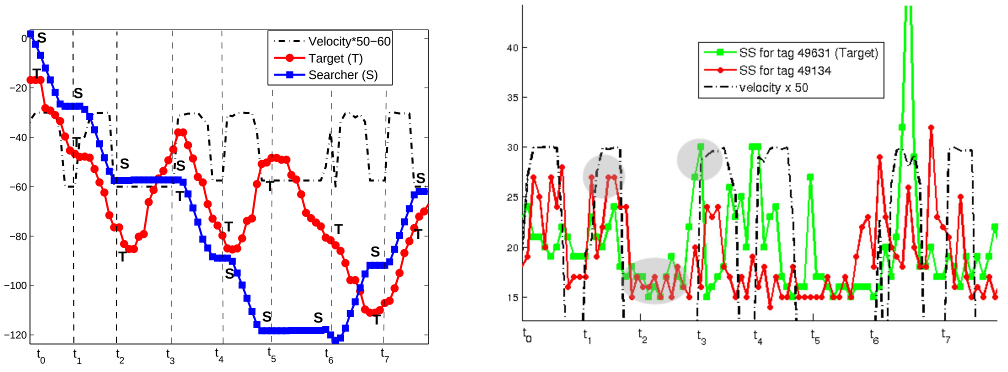
There will be four cases:

1. One particular frequency has a sharp rise in $\frac{f-m}{\sigma}$ while others do not. We declare detection for this frequency.

2. All frequencies have a sharp rise in $\frac{f-m}{\sigma}$. This case is a false positive because in a realistic situation we cannot differentiate between a possible detection from a nearby tag and the background noise.

3. No frequencies have a rise in $\frac{f-m}{\sigma}$. Then, that is a non-detection.

4. There is no rise in $\frac{f-m}{\sigma}$, but the tag is close-by. Then, this is a false negative. Since the antenna is directional, false negatives are caused by mis-aligned antenna.



(a) (b)

Figure 6.12: (a) The target's and the searcher's location along the corridor versus time. Here, the searcher and the target are denoted by T and S respectively. (b) The Signal Strength (SS) for target tag (49631 KHz) and an example frequency (49134 KHz) that does not correspond to a nearby tag. The mean level (m) for both frequencies is highlighted by the middle ellipse. The highlighted peak for 49134 is a false positive detection of tag 49134.

Fig. 6.11 depicts the signal strength for the frequencies associated with target (tag number 49611) and tag number 49124 that is not present in the trial. First, observe that the signal strength for both frequencies drops considerably as the robot stops and turns off its motor (zero speed intervals in Fig. 6.11).

Second, notice the peak in the strength for 49124 KHz marked by t_6. Observe that according to criterion in (6.3) we would declare a detection which is clearly wrong as no transmitters for 49124 KHz were present. Also, this makes the target detection at time t_4 an uncertain detection. This instance is in favor of our crossing model. That is, we must turn off the motor on a discrete set of nodes in order to reduce the noise interference from motor and avoid false positives. Similar false positives and negatives were observed to occur on both the ASV and AGV platforms.

6.2.4.2 Experiments with the AGV

In this experiment, the searcher and the target move along a corridor of length $120m$ while they have an offset of $20m$.

The target, carries the tag marked by 49631. Similar to the ASV case, the searcher sweeps the corridor from left to right. During this sweep, the searcher stops after uniform time intervals which are indicated by zero speed of the vehicle. Fig. 6.12(a) illustrates the location of the searcher and the target along the line as well as the waiting intervals of the searcher. Here, the searcher's strategy is an example of our proposed R^2SR^2S... strategy.

Fig. 6.12(b) depicts the signal strength received by the searcher from the tag as well as the frequency associated with tag number 49134 which is not present in the trial. First, observe the peaks in the signal strength from the target tag. Here, according to the distances in Fig. 6.12, the peaks at t_3, t_4 and $[t_6, t_7]$ are true detections. Notice that the searcher does not miss the target at the particular crossing event at $[t_6, t_7]$. Also, the detection at t_5, despite the large distance ($73m$), is because of complete alignment between the antenna and the target.

Observe from Fig. 6.12(b) the two highlighted peaks in the signal strength at $[t_1, t_2]$ and t_3. Notice that the peaks for both frequencies are of similar maximum value and are both higher than their corresponding mean level (which is also highlighted). Therefore, according to our criterion $\frac{f-m}{\sigma}$ (6.3), we would declare detection for these frequencies at t_3 and $[t_1, t_2]$ respectively. However, the detection for 49134 would be a false detection since no transmissions on this frequency were used in the trial. Notice that this false detection is received while the vehicle is moving and its motors are on. On the other hand, the peak from the target frequency corresponds to a true detection since the searcher and the target are very close (see Fig. 6.12). Moreover, the measurement is taken while the searcher's motors are off. To summarize, this example supports our discrete crossing model. That is, the searcher has to stop in order to take reliable measurements.

6.2.5 Summary

We studied the problem of searching for random walker on a set of discrete nodes while lie on a line segment. The goal is to design search strategies that maximize the probability of capturing the target subject to constraints on the available search time. The searcher's possible actions are move to left or right or stay on the current node.

We studied the crossing detection model, and formulated the problem as a POMDP.

We observed an interesting structure in the POMDP solutions: a groups of right actions interleaved with stay actions, i.e. $(R^kS)^m$. After analyzing the capture probability of these strategies as a function of k and m we showed that the best solution is $(R^2S)^m$. We then demonstrated the applicability of our proposed strategies in our carp monitoring project. We used ASVs and AGVs as our searcher robot and target. We showed that the crossing detection model is a good assumption for our directional sensors.

In the next section, we study search strategies to find a two-dimensional random walker.

6.3 Simple Random Walker on Two-Dimensional Grids

In this section, we study the problem of finding a target which is moving in a two-dimensional grid. In particular, the searcher and the target are moving in a square region. The searcher's goal is to detect the target which is possible when the target is within the searcher's capture range. As a result, we discretize the square into cells where the sides of each cell is proportional to the capture radius. Therefore, the environment is modeled as a $N \times N$ grid. The searcher and the target move in discrete time steps. We have capture when the searcher and the target are in the same cell at the same time.

At each time step the target moves to the left or to the right with probability h, up or down with probability v, and it stays at its current cell with probability s. We assume that the borders are reflective: If the target is about to leave the grid, it will reverse directions instead. For example, if the target is on the top edge of the grid, with probability $2h$ it moves down. Throughout this paper, we often assume that movement is uniform: $h = v = s = \frac{1}{5}$.

The searcher on the other hand may choose among what we will call *macro-actions*. Before executing a macro-action, the searcher will select a column. The macro-action itself consists of three stages. In the first stage the searcher will travel horizontally to the selected column. Then the searcher will wait until N time steps have passed. The final stage is to sweep across the column. Afterwards, the searcher chooses another macro-action. We implement the waiting period in order to ensure that all macro-actions take $2N$ time steps, which simplifies the decision process.

The search task is to design a sequence of at most T actions (macro-actions) for the searcher such that the probability of capturing the target is maximized. Specifically, the goal is to design a search trajectory (strategy) Γ such that:

$$\begin{aligned} \max_{\Gamma} P_c(\Gamma = \{A_1, A_2, \cdots, A_k\}) \quad s.t. \quad k \leq T \\ A_i \in \{\text{column}_1, \text{column}_2, \cdots, \text{column}_N\} \end{aligned} \tag{6.4}$$

where $P_c(\Gamma)$ represents the capture probability of Γ.

We denote the probability distribution describing the target's location on the grid at time t by X_t. This is the base distribution of the target, and it is independent of the searcher's strategy. In this section, we assume $X_1 = f$, where f is:

$$f(x) = \begin{cases} \frac{1}{(N-1)^2} & \text{if } x \text{ is in the interior of the grid} \\ \frac{1}{4(N-1)^2} & \text{if } x \text{ is in a corner of the grid} \\ \frac{1}{2(N-1)^2} & \text{otherwise} \end{cases} \tag{6.5}$$

for any location x on the grid. This assumption is justified by the following lemma:

Lemma 6.3.1. *Suppose $h > 0$, $v > 0$ and $s > 0$. For any distribution X_1 over the grid,*

$$\lim_{t\to\infty} X_t = f \tag{6.6}$$

Proof. This is true as a direct consequence of the Fundamental Theorem of Markov Chains. □

The distribution f is known as a *stationary distribution*:

Definition 16. *The Stationary Distribution, f, is the unique distribution over the grid such that if $X_1 = f$, then $X_t = f$ for all $t > 1$.*

We next present the formulation of our grid search problem as a MOMDP.

6.3.1 Approximation as a One-Dimensional Problem

We now present our approximation of the problem as a one-dimensional search problem which we formulate as a Mixed Observability Markov Decision Process (MOMDP). In the original two-dimensional problem, the target moves to the neighboring cells with equal probabilities. Using macro-actions, the target's motions is viewed only *between* the columns. As a result, there are N nodes to search for the target. Projecting the target's motion vertically, the target moves to the left or the right column with probability h and stays in its current column with probability $2v + s$. Therefore, in the one-dimensional setup the target's transition matrix for one time-step is obtained from a $N \times N$ tridiagonal matrix M where the values on the main diagonal are $2v + 1$, the values on the first diagonal below and above the main diagonal are h[5]. Since the length of the macro-actions is $2N$ we should use the transition probabilities after $2N$

[5] The reflective boundary properties should be taken care of the first and the last rows.

time-steps. Thus, the final transition matrix for the target in the one-dimensional setup is M^{2N} along a single macro-action.

In summary, the one-dimensional approximation is formalized as a MOMDP as follows. There are N nodes along a line. The searcher's set of actions (macro-actions) is to visit each of the N nodes. The state is represented by $(p_s, p_e, t, f_c,)$ where $p_s \in 1, 2, \cdots, N$ is the searcher's location, $p_e \in 1, 2, \cdots, N$ is the location of the target, $t \leq T$ is the amount of the time budget consumed so far, and f_c is the capture indicator variable. After performing action column$_k$ the location of the searcher p_s will change to i, and the available time budget is decreased by one. The location of the target p_e evolves according to the transition matrix M^{2N}. The aforementioned MOMDP model is depicted in Figure 6.13. In this figure, the state at time t is $s_t = (p_s, p_e, t, f_c)$. After applying action A_t the next state is $s_{t+1} = (p'_s, p'_e, t+1, f'_c)$.

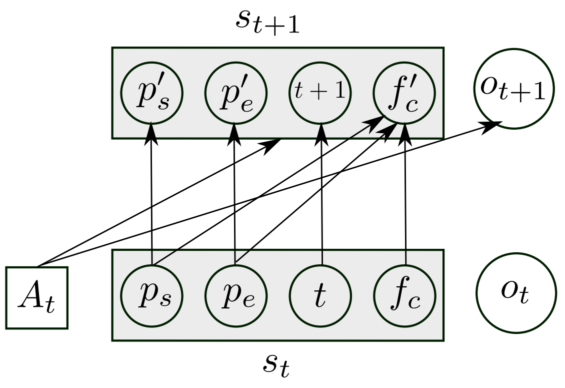


Figure 6.13: The MOMDP approximation of the problem.

In order to solve the optimization problem in (6.4) the reward function should be defined carefully. We are interested in finding a sequence of macro-actions Γ such that the capture probability is maximized:

$$\begin{aligned} P_c(\Gamma) = & P(\text{capture}|s_0, A_0) + \\ & P(s_1|s_0, A_0)P(\text{capture}|s_1, A_1) + \\ & P(s_1|s_0, A_0)P(s_2|s_1, A_1)P(\text{ capture}|s_2, A_2) + \cdots \end{aligned} \tag{6.7}$$

Notice that if we set the reward function for transitions to the capture state to one and all the remaining transitions to zero, the collected reward in our POMDP will be equal to the desired capture probability in (6.7).

We are interested in the capture probability of performing $A = \text{column}_k$ in a given state $(p_s, p_e, t) = (i, j, t)$.

Sub-problem 1. *Compute the probability that the searcher and the target in the $N \times N$ grid colocate in the same cell when the target starts somewhere in column j and the searcher performs the following strategy: The searcher starts from $(1, i)$ and moves toward $(1, k)$, waits at $(1, k)$ for $(N - |k - i|)$ steps and then moves along the k^{th} column from $(1, k)$ to (N, k).*

In order to get a handle on the problem above, we need to first provide some assumptions about the exact location of the target along column j. Here, we assume that it is equally likely that the target is in any row along the column j.

The probability in sub-problem 1 can be lower bounded by the probability of capture *only* when the searcher sweeps the k^{th} column. When the searcher is sweeping column_k it is at cell (r, k) at time $N + r$. Therefore, the problem of computing the capture probability reduces to counting the number of events that the target is at (r, k) at time $N + r$ given that the searcher has not captured the target earlier. An important observation here is that the only event that the searcher captures the target at time $N + r$ and also in an earlier time-step, namely at $N + r'$, is when the target moves straight up from (r, k) to (r', k). We can neglect this single event in computing the capture probability. Therefore, the sub-problem 1 is converted to:

Sub-problem 2. *Count the number of events that the target starts at (x, j) at time 0 and reaches at (r, k) at time $N + r$.*

Let n_u, n_d, n_r, n_l, n_s denote the number of time-steps that the target moves up, down, right, and left, and the number of actions that the searcher stays respectively. In fact, we should count the solutions to:

$$\begin{aligned} n_u - n_d &= r - x \\ n_r - n_l &= k - j \\ n_s + n_u + n_d + n_r + n_l &= N + r \end{aligned} \tag{6.8}$$

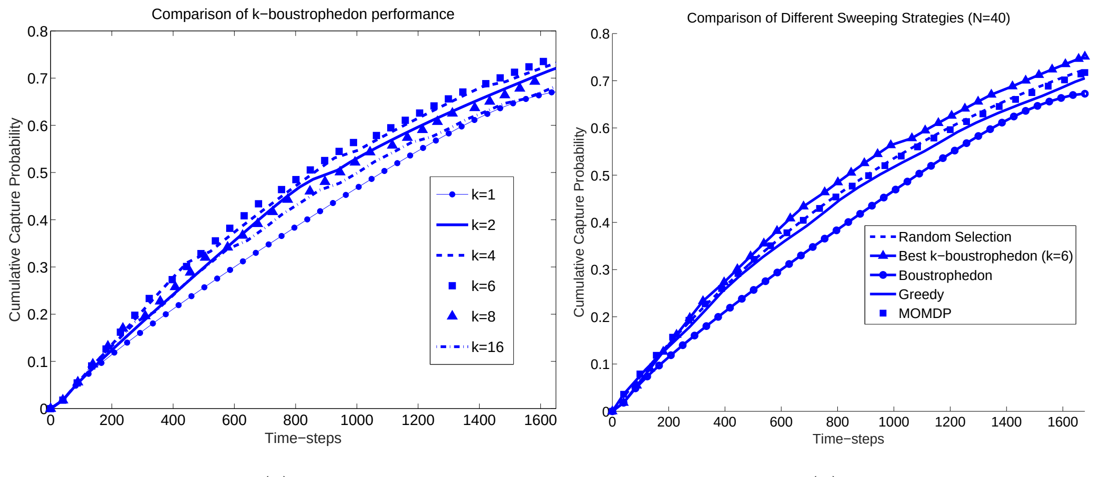
We now have all the ingredients for computing the MOMDP solutions. We use the Approximate POMDP Planning (APPL) toolkit which is available at [73,86]. Next, we present the results of our approximation.

6.3.2 Simulation Results

In this section we present the solutions of our proposed MOMDP strategies as well as the comparison of their performance against some heuristic strategies. We present the results for a grid of size 40×40. Let us first introduce the following sweeping strategies:

- Boustrophedon: Cover the grid column by column.
- K-boustraphedon: The searcher sweeps columns at horizontal distance k i.e. $1, k+1, \cdots$. When the right-most edge is hit, the search is restarted from the first column.
- Random Selection: The searcher picks a column at random, sets a timer to $2N$, sweeps the column and at the end of the sweep waits until timeout. The searcher repeats this strategy until the time budget is used.
- Greedy: In the greedy strategy at each iteration the searcher selects the column which will maximize the probability that the target will be captured.

To obtain the optimal solution for the MOMDP, we used the solver by [86].



(a) (b)

Figure 6.14: (a) Performance of k-boustrophedon for different ks. (b) Performance of different sweeping strategies.

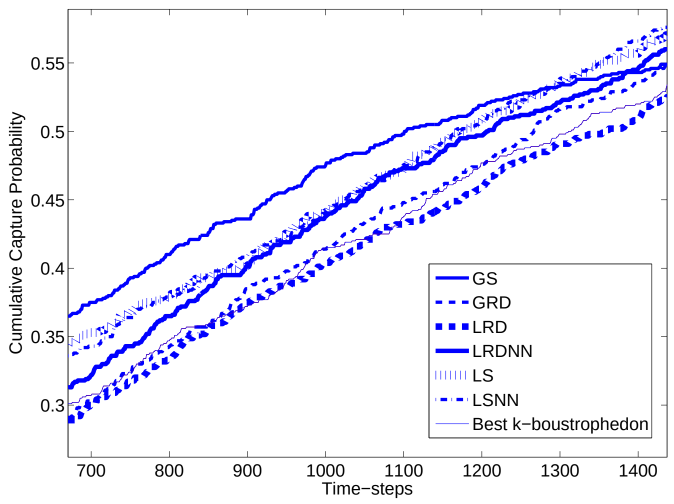


Figure 6.15: Comparison of the k-boustrophedon strategy ($k = 6$) with heuristic strategies.

We first compare the performance of k-boustraphedon strategies for different values of k. It turns out the best value is $k = 6$ (Fig. 6.14(a)).

Next, we compare the best k-boustraphedon strategy with the greedy, MOMDP, and random selection strategies. Figure 6.14(b) shows the cumulative capture probability versus time for these sweeping strategies. From Fig. 6.14(b) the best strategy is the k-boustrophedon candidate with $k = 6$. Next, we compare the k-boustrophedon strategy ($k = 6$) with the following heuristic strategies. Here, the strategies with "local" keyword are the ones that divide the region into smaller regions and perform a strategy in each of them.

- Global Random Direction (GRD): The searcher picks a random point on the boundary. Then moves there by first adjusting vertically and then horizontally.
- Local Random Direction (LRD): The searcher divides the region into smaller square regions, selects one of them at random and performs the Global Random Direction (GRD) strategy in each of them.
- Local Random Direction Not Neighbors (LRDNN): The searcher performs the LRD strategy such that the consequtive local regions are not adjacent.

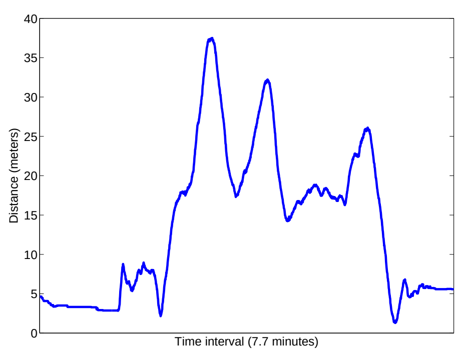
- Global Spiral (GS): The searcher moves along a square path, and repeats by increasing the side of the square.
- Local Spiral (LS): The searcher divides the region into smaller square regions; Selects one of them at random, and performs the GS strategy in each of them.
- Local Spiral Not Neighbors (LSNN): The searcher performs the LS strategy such that the consequtive local regions are not adjacent.

Fig. 6.15 shows the performance of the above heuristic strategies against our best sweeping strategy i.e. k-boustrophedon with $k = 6$. The strategies are very close in performance, which is an interesting observation: As long as there is "some" gap between the columns, the precise choice of column does not affect the performance drastically.

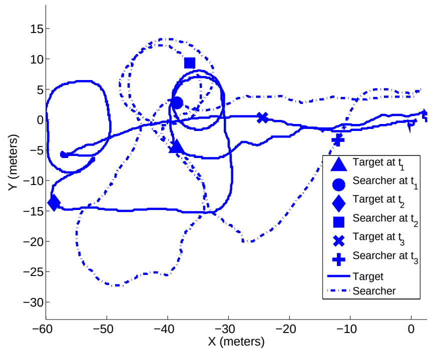
Next, we present our field experiments to show our proposed strategies in action.

6.3.3 Experimental Demonstration

In this section, we present our field experiments to demonstrate the applicability of our column sweeping strategies. A description of our system is given is Section 6.1.2. Similar to the linear graph case, in our experiments we use we use one ASV as the searcher robot while the other ASV plays the role of the fish which moves around randomly. We performed our experiments in Lake Staring, Minnesota. In order to keep the boats close to the shore so that we have control over them in case of GPS signal loss, we conducted the experiments in a square region of relatively small size, $40m \times 40m$. We discretized the region into cells of size $10m \times 10m$, and thus $N = 4$. We executed the greedy strategy. Figure 6.16(b) provides the GPS traces for the target and the searcher and Fig. 6.16(a) shows the distance between them. Here the target is programmed to perform an instance of a random walk.



(a)



(b)

Figure 6.16: (a) The distance between the searcher and the target during the trial is depicted. (b) The GPS traces for the searcher and the target are shown. The trajectories are marked at three different time instances t_1, t_2 and t_3.

6.3.4 Summary

In this section we studied search strategies to find a random walker on a grid. We restricted our attention to a specific class of strategies which we refer to as the sweeping strategies. These strategies are composed of macro-actions which are sweeping an entire column. As a result the problem is reduced to finding the best set of columns such that the detection probability is maximized.

We first presented the approximation of our problem as an MOMDP. Then we compared the performance of MOMDP solutions with some heuristic strategies such as greedy and random column selection. Perhaps the main take-away from our results is that sweeping strategies are robust to column selection. This makes sweeping strategies a good choice for robotics applications where disturbances, sensing and actuation uncertainties may make it hard to follow precise trajectories. Finally, we showed our field experiments to show the application of our sweeping strategies in the invasive fish monitoring projection.

6.4 Concluding Remarks

In this chapter, we studied the problem of searching for random walker on a discrete line segment and also a two-dimensional grid. The goal is to design search strategies that maximize the probability of capturing the target subject to constraints on the search time budget. In the case of the linear graph and crossing detection model, we showed that the best strategy is of the form $(R^2S)^m$. In the case of two-dimensional grids, we compared the performance of heuristic strategies such as random column selection with MOMDP solutions for the best set of columns. We showed that in simulation these strategies are similar in terms of detection probability. Finally we presented our preliminary experiments for application of our results to a practical problem, where the main objective is to find radio-tagged fish in a lake by using an autonomous surface/ground vehicle. After the description of the robotic system and its sensor model, we presented results from field experiments carried out in a Minnesota lake.

Chapter 7

Open Problems and Concluding Remarks

In this dissertation, we studied two types of search problems in robotics: adversarial and probabilistic search problems. The goal for adversarial search problems, which are known as pursuit-evasion games, was to capture an adversarial mobile target that is capable of performing the best possible strategy to escape capture. We studied versions of pursuit-evasion games on three-dimensional surfaces as well as in polygonal environments with limited sensing specifically line-of-sight visibility. Our results in this domain are in the form of capture strategies with guaranteed finite time capture.

In probabilistic search problems our goal was to find a mobile target which is moving according to a random walk motion model, and thus independent of the searcher's strategy. Our goal was to design a search strategy that maximizes the detection probability subject to the restrictions that the searcher has such as the time budget. Our results are near-optimal strategies with the presented capture probability.

In the following, we first summarize our contributions in both domains and also highlight some future research directions and open problems.

7.1 Summary of Contributions and Open Problems

In Chapters 4 and 5 we studied pursuit-evasion games while in Chapter 6 we studied probabilistic search problems for finding a random walker.

In Chapter 4, we showed that a single deterministic pursuer with line-of-sight visibility can capture an evader whose speed is equal to the pursuer's in any monotone polygon. It turns out that if we slightly relax the monotonicity constraint and consider the class of weakly monotone polygons[1], capture is no longer guaranteed. Fig. 7.1 shows a weakly monotone polygon in which the evader can escape forever.

It is worth mentioning that our proposed capture time in monotone polygons is improved by Berry et al. [47] who use rook strategy and show capture in strictly sweepable polygons which are a generalization of monotone polygons.

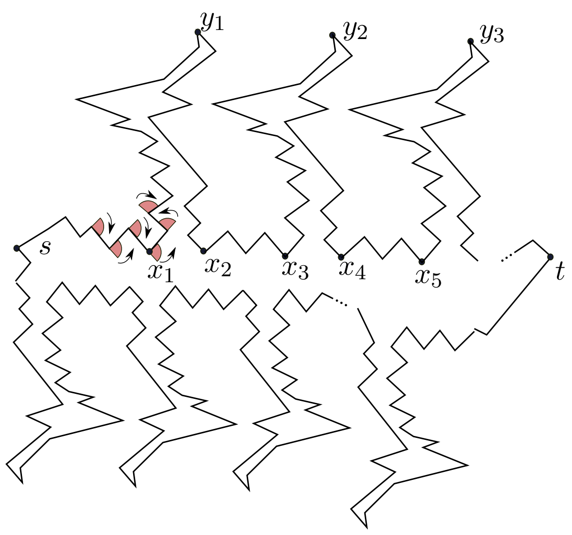


Figure 7.1: A weakly monotone polygon with respect to s and t. The upper chain that connects s to t is a repetition of the chain from s to y_4. The number of repetitions can be arbitrarily large. The chains from s to x_1, from x_2 to x_3, and from x_4 to x_5 are x-monotone chains. Also, the chains from x_1 to y_1, from y_1 to x_2, from x_3 to y_2 to x_4, from x_5 to y_3 are y-monotone chains. After disappearing from the pursuer's sight, the evader can hide in an upper or lower y-monotone polygon (whichever will be visited by the pursuer last) and escape when the evader is searching the other one.

In Chapter 5, we studied the lion and man game on polyhedral surfaces when the

[1] A simply connected polygon is weakly monotone with respect to vertices s and t if the following hold. Consider a particle that walks from s to t along the boundary in clockwise and counterclockwise directions. If in each of these walks, the range of the directions that the particle sweep does not include the negative x-axis, the polygon is weakly monotone with respect to s, t and x [87].

pursuer and the evader have complete knowledge of one another's location at all times. Our first result was the existence of a capture strategy with three lions and non-zero capture distance when the game is played on a general polyhedral surface in the presence of obstacles. One avenue for future research is to complement our existence result with an efficient algorithm to compute the strategy on a given polyhedron.

In Chapter 5, we also studied the lion and man game on convex terrains (height-maps). We presented a pursuit strategy which guarantees finite time capture when capture distance is non-zero. One of the questions left open is the optimality of our strategy in terms of capture time.

In Chapter 6, we studied the problem of searching for a one-dimensional random walker on a discrete line segment. Assuming the crossing detection model, using the structure of POMDP/MOMDP we showed that $(R^2S)^m$ is performing close to the strategies found by the MDP methods. We then studied the problem of finding a random walker on a two-dimensional grid. In order to tackle curse of dimensionality in the state space, we limited the form of the search strategies to sweeping strategies. In these strategies a set of columns is selected and each column is swept entirely. We approximated the problem of finding the best set of columns as a MOMDP. Then we compared the MOMPD solutions with some heuristic strategies such as boustrophedon. We concluded that these strategies are close to each other in terms of capture probability as long as the distance between the consecutive columns is not too small and not too large. We conjecture that the best separation is $\sqrt{N}$ in an $N \times N$ grid.

From application perspective, one interesting direction is to extend our experimental results for detecting radio-tagged fish. One approach is to define a more accurate model of the system. For example, the antenna might miss a tag (false negative) or it may report a tag mistakenly (false positive). In addition, the antenna is directional and so its detection region is more accurately modeled by an oval and not a circle. Another approach is designing strategies that are more robust to the conditions at the specific lake environment (such as wind) or the uncertainties in the platform (such as the sensor, the signal strength of the radio tags, and the underlying controller of ASVs).

7.2 Future Research Directions

This thesis presents fundamental results in pursuit-evasion games and probabilistic search problems. However, the open problems in the field are far more than the known results. We now present a list of open problems.

Continuous Time Model versus Turn-based Discrete Model

While in this dissertation we focused on the turn-based model, the techniques presented are applicable to the continuous-time version. It can be shown that any given turn-based capture strategy to get within distance r of the evader can be adopted to a continuous-time capture strategy with capture radius $r + s$ where s is the step size. (The continuous-time pursuer simply plays the discrete-time game with respect to the perceived previous location of the evader). Therefore, as long as the pursuer can change its step-size s, the capture guarantee for the continuous time model can get arbitrarily close to the capture guarantee in the turn-based model[2]. However, this argument is not applicable when planning must be performed in the configuration space. In fact, the problem is mostly open in the presence of non-holonomic constraints [88] on the motion of the players.

Environment Complexity

An interesting open question regarding pursuit-evasion games in two-dimensional setups is to determine the classes of polygons that are two-pursuer-win [89]. In addition, although pursuit-evasion problems in higher dimensions are very important from a practical perspective many questions remain unanswered. For example, while we know that three pursuers are sufficient for capture on general three-dimensional surfaces, the set of one-pursuer-win or two-pursuer-win surfaces are unknown. Similarly, determining the minimum number of pursuers to guarantee capture on terrains modeled as height maps is an open question.

[2] In the turn-based strategy, the time unit Δt can be chosen arbitrarily small. Consequently, the step-size s can be arbitrarily small since the players' speed is fixed.

Limited Visibility Pursuit-Evasion Games

Very few results are available regarding limited visibility pursuit-evasion games. In this dissertation we showed than single-pursuer line-of-sight visibility capture is guaranteed in monotone polygons, but we do not yet have a full characterization of the class of polygons that are pursuer-win. Berry et al. [47] describe a line-of-sight pursuit strategy when the game is played in strictly sweepable polygons, which are a generalization of monotone polygons. A polygon is *strictly sweepable* if a straight line can be moved continuously over the entire polygon (via a sequence of translations and rotations) such that (1) the intersection of the line and polygonal area is always a connected line segment, and (2) no point is swept more than once. In particular, a monotone polygon is the special case of sweeping the polygon via a single translation. An intermediate goal would be to determine whether *sweepable polygons* are pursuer win. In this family of polygons, the sweep line may visit points more than once.

Finally, there are no results regarding the problem with limited-sensing pursuers on polyhedral surfaces. Questions such as the minimum number of pursuers with line-of-sight visibility, and conversely the class of surfaces that are k-pursuer-win for a given k, are rich open problems.

Unmanned Autonomous Vehicles (UAVs) as the Searcher or the Target

In recent years Unmanned Autonomous Vehicles (UAVs) have received increasing attention. Interesting variants of the problem can be designed when the players have the ability to fly. For example, suppose that the evader is restricted to move on the ground which is modeled as a geodesic terrain while the pursuer is able to fly. What is the outcome of the game subject to limitations on the highest altitude accessible by the pursuer? Can we provide guarantees on capture when we have a heterogeneous team of flying and ground pursuers?

Different Motion Models

Finally, in all of the variants above different motion models can be considered for the evader. Depending on the application, we can choose between adversarial or various stochastic motion models. Another interesting extension is to consider various models for the movement of the pursuer. For example, Ruiz and Isler [90] show that a pursuer modeled as a Differential Drive Robot (DDR) captures an Omnidirectional evader in a convex environment. A full characterization of the lion and man game with car-like motion models in more complex environments remains unknown.

7.3 Final Remarks

The field of search and pursuit-evasion remains rich with lots of interesting applications in robotics. The everyday progress in robotic platforms, which makes the application of search problems practical, together with their theoretical soundness foster an important research direction for robotics community. In this dissertation, we studied fundamental problems in the field. However, many questions remain open each of which have the potential for years of research demanding the knowledge of different groups of researchers. Our hope is that researchers continue the progress in both the algorithmic and the technological aspects and robotic searchers become feasible in everyday applications.

References

[1] V. Isler, N. Noori, P. Plonski, A. Renzaglia, P. Tokekar, and J. Vander Hook. Finding and tracking targets in the wild: Algorithms and field deployments. In *International Symposium on Safety, Security, and Rescue Robotics*, 2015. To Appear.

[2] iRobot. `http://www.irobot.com`.

[3] Da Vinci surgery. `www.davincisurgery.com`.

[4] Magnetic microbots to fight cancer. `http://spectrum.ieee.org/robotics/medical-robots/magnetic-microbots-to-fight-cancer`.

[5] Google self-driving car project. `https://www.google.com/selfdrivingcar/`.

[6] N. Noori, A. Renzaglia, J. Vander Hook, and V. Isler. Constrained probabilistic search for a one-dimensional random walker. *IEEE Transaction on Robotics*, 2015. To Appear.

[7] N. Noori, A. Beveridge, and V. Isler. A pursuit-evasion toolkit. 2015.

[8] N. Noori and V. Isler. Lion and man with visibility in monotone polygons. In *Algorithmic Foundations of Robotics X*, volume 86 of *Springer Tracts in Advanced Robotics*, pages 263–278. Springer Berlin Heidelberg, 2013.

[9] A. Renzaglia, N. Noori, and V. Isler. Searching for a one-dimensional random walker: Randomized strategy with energy budget. In *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*, pages 6019–6024, Nov 2013.

[10] N. Noori, A. Renzaglia, and V. Isler. Searching for a one-dimensional random walker: Deterministic strategies with a time budget when crossing is allowed. In *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*, pages 4811–4816, Nov 2013.

[11] N. Noori and V. Isler. The lion and man game on polyhedral surfaces with obstacles. *In Review, Theoretical Computer Science*, 2015.

[12] N. Noori and V. Isler. Lion and man with visibility in monotone polygons. *The International Journal of Robotics Research*, 33(1):155–181, 2014.

[13] N. Noori and V. Isler. The lion and man game on convex terrains. Technical Report 14-020, Department of Computer Science & Engineering, University of Minnesota, 2014.

[14] N. Noori and V. Isler. The lion and man game on polyhedral surfaces with boundary. In *Intelligent Robots and Systems (IROS 2014), 2014 IEEE/RSJ International Conference on*, pages 1769–1774, Sept 2014.

[15] A. Renzaglia, N. Noori, and V. Isler. The role of target modeling in designing search strategies. In *Intelligent Robots and Systems (IROS 2014), 2014 IEEE/RSJ International Conference on*, pages 4260–4265, Sept 2014.

[16] N. Noori and V. Isler. The lion and man game on convex terrains. In *Workshop on the Algorithmic Foundations of Robotics (WAFR)*, 2014.

[17] F. V. Fomin, P. A. Golovach, J. Kratochvl, N. Nisse, and K. Suchan. Pursuing a fast robber on a graph. *Theoretical Computer Science*, 411(79):1167 – 1181, 2010.

[18] M. Adler, H. Racke, N. Sivadasan, C. Sohler, and B. Vocking. Randomized pursuit-evasion in graphs. *Combinatorics Probability and Computing*, 12(3):225–244, 2003.

[19] V. Isler and N. Karnad. The role of information in the cop-robber game. *Theor. Comput. Sci.*, 399(3):179–190, June 2008.

[20] K. Klein and S. Suri. Capture bounds for visibility-based pursuit evasion. In *Proceedings of the Twenty-ninth Annual Symposium on Computational Geometry*, SoCG '13, pages 329–338, New York, NY, USA, 2013. ACM.

[21] I. Suzuki and M. Yamashita. Searching for a mobile intruder in a polygonal region. *SIAM Journal on Computing*, 21(5):863–888, 1992.

[22] L. J. Guibas, J. Latombe, S. M. Lavalle, D. Lin, and R. Motwani. A visibility-based pursuit-evasion problem. *Internat. J. Comput. Geom. Appl.*, 9(4-5):471–493, 1999.

[23] T. Chung, G. Hollinger, and V. Isler. Search and pursuit-evasion in mobile robotics. *Autonomous Robots*, 31(4):299–316, 2011.

[24] C. Robin and S. Lacroix. Multi-robot target detection and tracking: taxonomy and survey. *Autonomous Robots*, pages 1–32, 2015.

[25] R. Nowakowski and P. Winkler. Vertex-to-vertex pursuit in a graph. *Discrete Mathematics*, 43(2–3):235 – 239, 1983.

[26] A. Quilliot. Jeux et pointes fixes sur les graphes. *Thése de 3éme cycle, Université de Paris VI*, pages 131–145, 1978.

[27] N. Megiddo, S. L. Hakimi, M. R. Garey, D. S. Johnson, and C. H. Papadimitriou. The complexity of searching a graph. *J. ACM*, 35(1):18–44, January 1988.

[28] M. Aigner and M. Fromme. A game of cops and robbers. *Discrete Applied Mathematics*, 8(1):1–12, 1984.

[29] A. S. Goldstein and E. M. Reingold. The complexity of pursuit on a graph. *Theoretical Computer Science*, 143(1):93 – 112, 1995.

[30] W.B. Kinnersley. Cops and robbers is exptime-complete. *Journal of Combinatorial Theory*, Series B(111):201–220, 2015.

[31] A. Kehagias, D. Mitsche, and P. Pralat. Cops and invisible robbers: The cost of drunkenness. *Theoretical Computer Science*, 481(0):100–120, 2013.

[32] J. E. Littlewood. *A mathematician's miscellany / J. E. Littlewood.* Methuen, London :, 1953.

[33] J. Sgall. Solution of David Gale's lion and man problem. *Theoretical Computer Science*, 259(1–2):663 – 670, 2001.

[34] L. Alonso, A. S. Goldstein, and E. M. Reingold. 'Lion and Man' : Upper and Lower Bounds. *Informs Journal on Computing*, 4:447–452, 1992.

[35] V. Isler, S. Kannan, and S. Khanna. Randomized pursuit-evasion in a polygonal environment. *Robotics, IEEE Transactions on*, 21(5):875–884, Oct 2005.

[36] A. Beveridge and Y. Cai. Two dimensional pursuit-evasion in a compact domain with piecewise analytic boundary. http://arxiv.org/abs/1505.00297.

[37] Z. Zhou, J. R. Shewchuk, H. Huang, and C. J. Tomlin. Smarter lions: Efficient full-knowledge pursuit in general arenas. Last accessed: December 13, 2015, 2012.

[38] D. Bhadauria, K. Klein, V. Isler, and S. Suri. Capturing an evader in polygonal environments with obstacles: The full visibility case. *The International Journal of Robotics Research*, 31(10):1176–1189, 2012.

[39] S. Bhattacharya and S. Hutchinson. On the existence of nash equilibrium for a two-player pursuit-evasion game with visibility constraints. *The International Journal of Robotics Research*, 29(7):831–839, 2010.

[40] S. Kopparty and C. V. Ravishankar. A framework for pursuit evasion games in Rn. *Information Processing Letters*, 96(3):114–122, 2005.

[41] S. D. Bopardikar and S. Suri. k-capture in multiagent pursuit evasion, or the lion and the hyenas. *Theor. Comput. Sci.*, 522:13–23, February 2014.

[42] S. Alexander, R. Bishop, and R. Ghrist. Pursuit and evasion in non-convex domains of arbitrary dimensions. In *Robotics: Science and Systems*, 2006.

[43] K. Klein and S. Suri. Pursuit-evasion on polyhedral surfaces. In *Algorithms and Computation*, volume 8283 of *Lecture Notes in Computer Science*, pages 284–294. Springer Berlin Heidelberg, 2013.

[44] L. J. Guibas, J. C. Latombe, S. M. Lavalle, D. Lin, and R. Motwani. A visibility-based pursuit-evasion problem. *Internat. J. Comput. Geom. Appl.*, 9(4-5):471–493, 1999.

[45] K. Klein and S. Suri. Catch me if you can: Pursuit and capture in polygonal environments with obstacles. In *26th Annual Conference on Artificial Intelligence (AAAI '12*, 2012.

[46] K. Klein and S. Suri. Capture bounds for visibility-based pursuit evasion. In *Proceedings of the Twenty-ninth Annual Symposium on Computational Geometry*, SoCG '13, pages 329–338, New York, NY, USA, 2013. ACM.

[47] L. Berry, A. Beveridge, J. Butterfield, V. Isler, Z. Keller, A. Shine, and J. Wang. Line-of-sight pursuit in strictly sweepable polygons. *CoRR*, abs/1508.07603, 2015.

[48] S. D. Bopardikar, F. Bullo, and J. P. Hespanha. On discrete-time pursuit-evasion games with sensing limitations. *Robotics, IEEE Transactions on*, 24(6):1429–1439, Dec 2008.

[49] N. Karnad and V. Isler. Bearing-only pursuit. In *Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on*, pages 2665–2670, May 2008.

[50] S. M. Pollock. A simple model of search for a moving target. *Operations Research*, 18(5):883–903, 1970.

[51] J. G. Wilson. On optimal search plans to detect a target moving randomly on the real line. *Stochastic Processes and their Applications*, 20(2):315 – 321, 1985.

[52] J. M. Dobbie. A two-cell model of search for a moving target. *Operations Research*, 22(1):79–92, 1974.

[53] Y. C. Kan. Optimal search of a moving target. *Operations Research*, 25(5):864–870, 1977.

[54] K. E. Trummel and J. R. Weisinger. The complexity of the optimal searcher path problem. *Operations Research*, 34(2):324–327, 1986.

[55] T.J. Stweart. Experience with a branch-and-bound algorithm for constrained searcher motion. *Search Theory and Applications*, 8:247–253, 1980.

[56] A. R. Washburn. Branch and bound methods for a search problem. *Naval Research Logistics (NRL)*, 45(3):243–257, 1998.

[57] G. Hollinger, S. Singh, J. Djugash, and A. Kehagias. Efficient multi-robot search for a moving target. *The International Journal of Robotics Research*, 28(2):201–219, 2009.

[58] S. Ong, W. Shao, D. Hsu, and W. Lee. Planning under uncertainty for robotic tasks with mixed observability. *The International Journal of Robotics Research*, 29(8):1053–1068, 2010.

[59] H. Lau, S. Huang, and G. Dissanayake. Probabilistic search for a moving target in an indoor environment. *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3393–3398, Oct 2006.

[60] F. Bartumeus, MGE. da Luz, GM. Viswanathan, and J. Catalan. Animal search strategies: a quantitative random-walk analysis. *Ecology*, 86(11):3078–3087, 2005.

[61] S. Redner. *A Guide to First-Passage Processes.* University Press, Cambridge, 1st edition, 2001.

[62] L. Lovász. Random walks on graphs: A survey. *Combinatorics, Paul Erdös is eighty*, 2(1):1–46, 1993.

[63] J.K. Anlauf. Asymptotically exact solution of the one-dimensional trapping problem. *Physical review letters*, 52(21):1845–1848, 1984.

[64] S. Redner and PL Krapivsky. Capture of the lamb: Diffusing predators seeking a diffusing prey. *American Journal of Physics*, 67:1277–1283, 1999.

[65] A. Gabel, S. N. Majumdar, N. K. Panduranga, and S. Redner. Can a lamb reach a haven before being eaten by diffusing lions? *Journal of Statistical Mechanics: Theory and Experiment*, 2012(05), 2012.

[66] R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction.* Adaptive Computation and Machine Learning Series. Mit Press, 1998.

[67] L. Kaelbling, M. Littman, and A. Cassandra. Planning and acting in partially observable stochastic domains. *Artificial Intelligence*, 101(1–2):99 – 134, 1998.

[68] S. Guy, J. Pineau, and R. Kaplow. A survey of point-based pomdp solvers. *Autonomous Agents and Multi-Agent Systems*, 27(1):1–51, 2013.

[69] J. Pineau, G. Gordon, and S. Thrun. Point-based value iteration: An anytime algorithm for pomdps. In *Proceedings of the 18th International Joint Conference on Artificial Intelligence*, IJCAI'03, pages 1025–1030, San Francisco, CA, USA, 2003. Morgan Kaufmann Publishers Inc.

[70] D. Silver and J. Veness. Monte-carlo planning in large pomdps. In *Advances in Neural Information Processing Systems 23*, pages 2164–2172. Curran Associates, Inc., 2010.

[71] H. Kurniawati, D. Hsu, and W. S. Lee. Sarsop: Efficient point-based pomdp planning by approximating optimally reachable belief spaces. In *Robotics: Science and Systems*, volume 2008. Zurich, Switzerland, 2008.

[72] A. Somani, N. Ye, D. Hsu, and W. S. Lee. Despot: Online pomdp planning with regularization. In *Advances In Neural Information Processing Systems*, pages 1772–1780, 2013.

[73] Approximate POMDP Planning (APPL) toolkit. `http://bigbird.comp.nus.edu.sg/pmwiki/farm/appl`. Accessed March, 2015.

[74] M. De Berg, O. Cheong, and M. van Kreveld. *Computational geometry: algorithms and applications*. Springer, 2008.

[75] V. Isler, S. Kannan, and S. Khanna. Randomized pursuit-evasion in a polygonal environment. *IEEE Transactions on Robotics*, 21(5):875–884, 2005.

[76] L. Guibas, J. Hershberger, D. Leven, M. Sharir, and R. Tarjan. Linear-time algorithms for visibility and shortest path problems inside triangulated simple polygons. *Algorithmica*, 2:209–233, 1987.

[77] J. Sgall. Solution of david gale's lion and man problem. *Theor. Comput. Sci.*, 259:663–670, May 2001.

[78] M. de Berg, O. Cheong, M. Van Kreveld, and M. Overmars. *Computational Geometry: Algorithms and Applications*. Springer-Verlag TELOS, Santa Clara, CA, USA, 3rd ed. edition, 2008.

[79] S. Eidenbenz, C. Stamm, and P. Widmayer. Inapproximability results for guarding polygons and terrains. *Algorithmica*, 31(1):79–113, 2001.

[80] Oceanscience. `http://www.oceanscience.com/`.

[81] P. Tokekar, E. Branson, J. Vander Hook, and V. Isler. Coverage and active localization for monitoring invasive fish with an autonomous boat. *IEEE Robotics and Automation Magazine*, 30(3):33–41, 2013.

[82] Clearpath robotics. `http://www.clearpathrobotics.com/`.

[83] S. Thrun, W. Burgard, and D. Fox. *Probabilistic robotics*. MIT press Cambridge, MA, 2005.

[84] MDP MATLAB toolbox. `http://www.inra.fr/mia/T/MDPtoolbox`. Accessed November, 2012.

[85] P. G. Bajer and P. Sorensen. Recruitment and abundance of an invasive fish, the common carp, is driven by its propensity to invade and reproduce in basins that experience winter-time hypoxia in interconnected lakes. *Biological Invasions*, 12(5):1101–1112, 2010.

[86] S. C. W. Ong, S. W. Png, D. Hsu, and W. S. Lee. Planning under uncertainty for robotic tasks with mixed observability. *The International Journal of Robotics Research*, 29(8):1053–1068, 2010.

[87] P. J. Heffernan. Linear-time algorithms for weakly-monotone polygons. *Computational Geometry*, 3(3):121 – 137, 1993.

[88] Z. Li and J. F. Canny. *Nonholonomic motion planning*, volume 192. Springer Science & Business Media, 2012.

[89] B. P. W. Ames, A. Beveridge, R. Carlson, C. Djang, V. Isler, S. Ragain, and M. Savage. A leapfrog strategy for pursuit-evasion in a polygonal environment.

International Journal on Computational Geometry and Applications, 25:77–100, 2015.

[90] U. Ruiz and V. Isler. Capturing an omnidirectional evader in convex environments using a differential drive robot. *IEEE Robotics and Automation Letters*, 2016. To Appear.